ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA CREACIÓN DE INTERFACES DE USUARIO UTILIZANDO EL PARADIGMA DE DIAGRAMAS A MANO ALZADA"

TESIS DE GRADO

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN ESPECIALIZACIÓN SISTEMAS MULTIMEDIA

Presentada por:

GONZALO GABRIEL MÉNDEZ COBEÑA
JAVIER ALEJANDRO TIBAU BENÍTEZ

Guayaquil - Ecuador

2009

AGRADECIMIENTO

A Dios.

A nuestra familia,
que siempre nos ha apoyado.

A Xavier Ochoa y Katherine Chiluiza,
quienes nos motivaron a llegar a la meta.

A nuestros amigos,
con quienes compartimos nuestra vida universitaria.

DEDICATORIA

Con mucho aprecio,
a nuestra familia y amigos,
por compartir nuestro sueño
y acompañarnos en nuestro camino.

TRIBUNAL DE GRADO

PRESIDENTE

Ing. Holger Cevallos Ulloa

DIRECTOR DE TESIS

Dr. Xavier Ochoa Chehab

MIEMBROS PRINCIPALES

Ing. Ana Tapia Rosero

Dra. Katherine Chiluiza García

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de esta Tesis de Grado, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la Escuela Superior Politécnica del Litoral”

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Gonzalo Gabriel Méndez Cobeña

Javier Alejandro Tibau Benítez

RESUMEN

La solución propuesta en el presente trabajo es una herramienta de prototipado que permite a los diseñadores bosquejar una interfaz de usuario de manera rápida y sencilla utilizando la pantalla táctil de una Tablet PC.

En el capítulo 1 se introducen los conceptos relacionados al prototipado. Se consideran además ciertas definiciones planteadas por las distintas ramas de la Ingeniería en Computación y aportes teóricos de varios autores con el propósito de introducir el marco previo para que permita definir con claridad los objetivos y el alcance de la solución a ser construida.

El capítulo 2 comienza puntualizando el problema que se busca resolver y plantea los lineamientos y características de la solución a ser implementada. Este capítulo continúa con una revisión de varias de las tecnologías actuales que atacan los problemas relacionados a la creación de prototipos a mano alzada. Los paradigmas de interacción, el reconocimiento de trazos y algunas de las soluciones existentes para la creación de prototipos, de baja fidelidad, son detallados a fin de plantear una visión general del marco teórico sobre el cual se trabajará en las secciones posteriores.

En el capítulo 3 se describen el Análisis y Diseño del sistema. Se exponen las diferentes decisiones de diseño tomadas como resultado del análisis

previo y se puntualizan los casos de uso, los módulos y componentes del sistema. El diseño lógico, de interacción con el usuario y las pruebas que se realizarán más adelante son también revisados en este capítulo.

Los detalles de implementación, tales como la plataforma utilizada y los dispositivos de hardware, se describen en el capítulo 4. Más adelante, se especifican las pruebas de usabilidad y satisfacción del sistema que se realizaron. Los resultados de los parámetros medidos en dichas pruebas se exponen en detalle en la parte final de este capítulo.

Finalmente, se especifican las conclusiones, recomendaciones de este trabajo y sugerencias para el mejoramiento de esta herramienta o diseños similares para su utilización en futuras aplicaciones.

INDICE GENERAL

INDICE DE FIGURAS

INTRODUCCIÓN

En las primeras etapas del desarrollo de software, muchas de las actuales herramientas para construcción y diseño de prototipos de Interfaces gráficas resultan con frecuencia, más que un beneficio, un inconveniente para el equipo del proyecto. Estas herramientas a menudo fuerzan a los diseñadores a especificar más detalles de los que se dispone y se necesita en la etapa de levantamiento de requerimientos, donde una de las tareas principales consiste en crear prototipos no funcionales y de baja fidelidad que sirvan a lo largo del proceso de identificación de las necesidades del cliente, usuarios finales y demás involucrados en el proyecto.

Una ventaja de crear prototipos es que éstos modelan el dominio específico de la aplicación sin considerar los aspectos técnicos en gran detalle. Su objetivo principal es hacer de una solución una propuesta más comunicable. Sin embargo, las herramientas para su creación disponibles en el mercado no ofrecen las condiciones necesarias para lograr una coordinación efectiva entre las ideas de los clientes y la de los ingenieros de requerimientos.

Este impedimento obedece, mayormente, a las diferencias inherentes a las correspondientes áreas de trabajo de las partes involucradas. Así, por una parte, los ingenieros y desarrolladores, suelen ver al prototipado como un proceso en el que no tienen lugar mayores avancen de implementación, dado que tienen la percepción de que la escritura de código fuente es una parte

esencial de su trabajo. Mientras que en otra perspectiva, los interesados en el sistema tienden a esperar que los prototipos ejecuten las funcionalidades que aparentan, lo cual, en las etapas iniciales del desarrollo de un proyecto es virtualmente imposible.

Resulta evidente la necesidad de aplicaciones que puedan ser usadas durante la etapa de levantamiento de requerimientos, en especial durante la creación de los prototipos de interfaces de usuario. Aplicaciones que permitan unificar las ideas generadas en una reunión entre clientes y usuarios finales y el equipo de desarrollo de software. Una de las estrategias que siguen muchos ingenieros de requerimientos, en lugar de utilizar software especializado, prefieren construir sus diseños como simples bosquejos que sugieran la apariencia de las interfaces que implementarán las funcionalidades solicitadas.

El presente trabajo de tesis describe el análisis, diseño y las funcionalidades implementadas en la herramienta de prototipado a mano alzada PenGUIn Prototyper (Pen-based GUI Prototyper), una herramienta interactiva que permite a los diseñadores bosquejar una interfaz de manera rápida y sencilla utilizando la pantalla táctil de una Tablet PC, brindando la versatilidad que existe al hacerlo en una pieza de papel y un lápiz, pero proveyendo a la interfaz construida de otras funcionalidades inherentes al soporte que los dispositivos táctiles brindan al usuario al anotar ideas y trazar gráficos que pueden ser luego comunicados a otros colaboradores.

CAPÍTULO I.

1 LOS PROTOTIPOS DE SOFTWARE Y SUS PROBLEMAS

En el presente capítulo se introducen los conceptos relacionados al prototipado de software. Se consideran además ciertas definiciones planteadas por las distintas ramas de la Ingeniería en Computación y aportes teóricos de varios autores con el propósito de introducir el marco previo para definir los objetivos y el alcance de la solución a ser construida. Al final del capítulo se plantea el problema a enfrentar y la justificación para el desarrollo de este proyecto de tesis.

1.1 Antecedentes: Los Prototipos de Software

Durante la etapa de levantamiento de requerimientos para la construcción de un sistema, la comunicación con los clientes, usuarios finales y demás interesados en el desarrollo de dicho sistema es un factor vital a fin de determinar los objetivos finales que persigue el proyecto. Bajo esta premisa, elementos como la frecuencia de reuniones, los métodos de comunicación elegidos y los mecanismos que se escojan como medio de presentación de avances deben ser pensados de tal forma que maximicen la cantidad

de información que puede ser extraída en los diferentes encuentros con los interesados.

Una de estas técnicas, empleadas como medio de presentación de avances del proyecto a la parte interesada, es la exposición de prototipos del sistema en construcción. Los prototipos son utilizados en las ramas afines a la Ingeniería en Computación, por ejemplo, para esquematizar las interfaces de usuario que compondrán un sistema en particular. En estos casos, el prototipado ayuda a generar ideas acerca de cómo debe lucir una interfaz de un producto, lo que implica plasmar su futura apariencia con el propósito de aprender más acerca de lo que se espera de ésta mucho antes que el ciclo de desarrollo comience [2], es decir, mientras aún se mantienen abiertas las opciones de implementación.

1.2 Utilidad de los Prototipos

La elaboración de prototipos es una técnica utilizada en muchas ramas de Ingeniería y Ciencias en general. Etimológicamente, la palabra prototipo proviene de los vocablos griegos *protos* y *typo* que significan primero y forma o figura, respectivamente. Así, un prototipo puede ser definido como un ejemplar o modelo que combina los atributos más representativos de una clase o categoría.

Generalmente sirve como punto de referencia frente a la

incertidumbre existente al trabajar en un nuevo diseño, pues permite a los ingenieros y diseñadores explorar diferentes alternativas de solución, ejecutar pruebas y conocer ciertos parámetros de desempeño mucho antes de la puesta en producción del nuevo producto.

Mediante la presentación de prototipos es posible evaluar la calidad y hacer ajustes de una solución en las etapas iniciales de su desarrollo [1, 3] y es una técnica muy adecuada y recomendada en el desarrollo de software incremental o evolutivo [1], donde se planifican varias versiones entregables del sistema antes de la versión final.

El prototipado efectivo puede, por tanto, mejorar la satisfacción del usuario, ayudar a clarificar los requerimientos, el alcance, reducir el tiempo, los recursos y las correcciones necesarias a lo largo del desarrollo de un producto.

1.3 El Prototipado

En las siguientes secciones se detallarán las definiciones teóricas sobre los cuales se fundamentarán los objetivos del presente proyecto de tesis en cuanto a la creación de prototipos de interfaces de usuario. Ciertas definiciones planteadas por las distintas ramas de la Ingeniería en Computación son consideradas y, en algunos casos, replanteadas a través del aporte de varios autores. Este contenido

teórico introducirá el marco previo para definir con claridad los objetivos, y más adelante, el alcance de la solución a ser construida en este trabajo y que permita resolver el problema planteado al final del capítulo.

1.3.1 Enfoques del Prototipado

La Ingeniería de Software define dos enfoques principales para el prototipado: El Prototipado Evolutivo y el Prototipado “Throw-away” [7]. Ambos enfoques tienen objetivos específicos. En el primer caso, se busca entregar a los usuarios un sistema funcional y su desarrollo comienza con aquellos requerimientos que son mejor comprendidos por el equipo. Por otra parte, el prototipado “Throw-away” tiene como objetivo validar u obtener los requerimientos de un sistema en sus etapas iniciales. En este segundo enfoque, el proceso comienza con las partes menos comprendidas del sistema.

Una visión más focalizada para la clasificación de los enfoques del prototipado es propuesta por D. Bäumer et al. [1], donde se definen tres enfoques de prototipado:

- **Prototipado Exploratorio.-** Sirve para clarificar los requerimientos y las potenciales soluciones. Permite generar discusiones acerca de qué debería lograr una tarea y la forma

en que ésta puede ser soportada por la tecnología disponible.

- **Prototipado Experimental.-** Se enfoca en realización de algunos requerimientos seleccionados previamente. Este enfoque permite dimensionar la idoneidad y viabilidad de una propuesta de diseño/implementación particular y sus resultados son usualmente prototipos funcionales.
- **Prototipado Evolutivo.-** Es un proceso continuo para adaptar una aplicación a restricciones y limitantes organizacionales que cambian rápidamente. Aunque este enfoque puede resultar en cualquier clase de prototipos, tiene particular importancia cuando se desea construir sistemas pilotos.

1.3.2 Clases de Prototipos de Interfaces de Usuario

Además de definir una clasificación de los posibles enfoques de prototipado de software, los objetivos del presente proyecto de tesis y el trabajo a realizarse serán definidos teniendo en cuenta la siguiente clasificación de prototipos de interfaces de usuario. La primera clasificación está dada en función de los propósitos y el nivel de funcionalidad implementada que tiene un prototipo:

- **Prototipos de Presentación.-** Son construidos para ilustrar la forma en que una solución puede resolver los requerimientos

dados. Como son usados a menudo como parte de la propuesta de un proyecto, tienen un fuerte, y casi exclusivo, enfoque en la interfaz de usuario (apariencia del sistema).

- **Prototipos Funcionales.-** Implementan estratégicamente partes importantes tanto de la interfaz de usuario como de la funcionalidad de la aplicación planeada.
- **Sistemas pilotos.-** Son prototipos que incluyen la implementación de la mayoría de los requerimientos del sistema y son lo suficientemente robustos como para ser, prácticamente, puestos en producción.

Por otra parte, considerando la exactitud con la que un prototipo es capaz de *representar* la apariencia de un sistema, pueden clasificarse en [8]:

- **Prototipos de Baja Fidelidad.-** Son rápidos, baratos y diseñados para información del usuario lo más pronto posible.
- **Prototipos de Alta Fidelidad.-** Son más costosos que los anteriores e implican generalmente un trabajo de programación adicional. Sin embargo, se aproximan más al diseño final que tendrá el sistema.

1.3.3 Técnicas y Herramientas de Prototipado

Independientemente del enfoque de prototipado a seguir en el desarrollo de un sistema, resulta importante el hecho de que las personas que manipularán el prototipo construido tengan una visión general del futuro funcionamiento y apariencia del sistema que éste intenta describir.

En el mundo de la industria, existe una necesidad clara de prototipado que permita la pronta validación de los requisitos con el usuario. Por otra parte, este prototipado debe ser lo más completo posible para que la validación realizada por el usuario sea útil. Estos dos puntos se contraponen, ya que si se prototipa rápidamente, el prototipo es incompleto, pero si se crea un prototipo completo, éste no puede hacerse en un corto periodo de tiempo [6].

Varias técnicas de prototipado han surgido como propuestas para la creación de prototipos de interfaces gráficas. El surgimiento de estas técnicas ha motivado además la aparición de herramientas especializadas en soporte para la creación de prototipos de interfaces gráficas, muchas de las cuales ofrecen además funcionalidades avanzadas como pruebas de usabilidad, generación de código fuente, entre otras.

Sin embargo, el uso de herramientas de este tipo implica una

inversión no sólo en términos de recursos económicos (costo), sino también entrenamiento y capacitación de personal. Así, su principal uso resulta evidente en escenarios donde es requerida la construcción de Prototipos de Alta Fidelidad, es decir, donde se procura simular el producto final tanto como sea posible en términos de *layout* (distribución de componentes), colores, tamaños, etc.

En contraste, los Prototipos de Baja Fidelidad, presentan muchas más facilidades que pueden aprovecharse al momento de su construcción. Dado que estos se enfocan únicamente en conceptos claves tales como el conjunto de controles que serán utilizados y el orden en el cual se ejecutarán las operaciones provistas por la interfaz, es posible desarrollarlos con herramientas tan básicas y comunes como el papel y lápiz [2]. Cuando el prototipado de interfaces de usuario utiliza de forma casi exclusiva estos dos recursos, se conoce como Prototipado de Bocetos Trazados a Mano o Hand-sketched Prototyping.

1.3.4 Bocetos Trazados a Mano

El prototipado de bocetos trazados a mano es el más simple de las técnicas para construcción de prototipos de baja fidelidad. Deriva en la creación de una imagen palpable acerca de lo que será una futura aplicación y su creación como su manipulación resulta rápida y muy

versátil. Constituye un excelente mecanismo para que los usuarios imaginen lo que será la futura aplicación sin necesidad de considerar en detalle aspectos técnicos (plataformas, ancho de banda, tiempo de respuesta, etc.) o gráficos (colores, tipografía, etc.). Este tipo de prototipado se usa en la etapa más inicial del diseño, a menudo incluso antes de determinar muchos aspectos del análisis de requerimientos, con la finalidad de recoger las primeras impresiones de los interesados.

Por las características explicativas del papel, los bocetos construidos son ideales para obtener retroalimentación e iniciar una temprana discusión con el cliente acerca de las ventajas y desventajas del diseño de una interfaz. Así, se vuelven muy útiles para comunicar las ideas que resultan de una conversación o entrevista.

Una de las principales ventajas de los bocetos es su velocidad de producción: En promedio, es posible realizar uno en quince o veinte segundos [12], de manera que se pueden generar gran cantidad de ellos en muy poco tiempo.

A continuación se detallan las principales ventajas y desventajas [12] del prototipado de bocetos trazados a mano:

Ventajas

- Pueden ser fácilmente construidos y usados en etapas muy tempranas del proceso de desarrollo donde algunos problemas de usabilidad pueden ser identificados y corregidos rápidamente.
- Su construcción no requiere de habilidades de programación o conocimiento previo, ni grandes inversiones de tiempo o dinero.
- Son fácilmente portables, lo cual disminuye las limitantes al momento de realizar presentaciones o pruebas.
- Su modificación resulta bastante sencilla y puede ser realizada tantas veces como sea necesario.
- Son más fáciles de entender que otras especificaciones formales, principalmente para los usuarios no técnicos.
- Dado que son aproximaciones, a menudo producen más sugerencias y comentarios críticos que los prototipos de apariencia más real.

Desventajas

- Ofrecen sólo aproximaciones bastante rústicas de la interfaz; por lo que, por ejemplo, no es posible aprovechar el valor semántico provisto por el uso de colores.

- Presentan limitaciones al ayudar a entender el sistema de navegación y el flujo.
- Su uso puede estar limitado a demostraciones dirigidas por un facilitador, donde los usuarios asumen un rol más pasivo.
- En el caso de sistemas bastante extensos, están restringidos únicamente a la especificación de las tareas más comunes.

La construcción de bocetos trazados a mano presenta además varias ventajas sobre algunos prototipos construidos en herramientas especializadas:

- Cero esfuerzo codificando.- Aún cuando es posible construir en poco tiempo una interfaz con apariencia bastante cercana al diseño final en herramientas como Visual Basic [10] o Dreamweaver [11], escribir código para hacer que la interfaz responda apropiadamente a las entradas provistas por un usuario pueden tomar tiempo. Con un bosquejo construido a mano, el tiempo de codificación es cero. Aún cuando se conozca muy bien el uso de una herramienta especializada, es muy difícil ser más rápido que eso.
- Evitan retroalimentación no deseada.- Una interfaz de apariencia casi real podría sugerir una clase equivocada de

retroalimentación por parte de los usuarios. Esta clase de prototipos evita que los usuarios centren su atención en aspectos irrelevantes como el color de las sombras o la alineación de los controles, ya que es obvio que la apariencia final de la interfaz no ha sido aún trabajada. Esto último sugiere a los usuarios enfocarse en los conceptos y la funcionalidad representados por el prototipo en evaluación.

- Estimulan la creatividad.- Nuestros cerebros responden más creativamente a las cosas que lucen no del todo terminadas. Y los usuarios, especialmente los no técnicos, son a menudo menos intimidados por los prototipos de papel que por las computadoras, de tal manera que se sienten más cómodos explorando el diseño propuesto.

La figura 1.1 muestra el esquema de una interfaz realizada en papel. El uso de bocetos o bosquejos trazados a mano permiten ilustrar los componentes de una futura interfaz, pero carecen de ventajas al momento de realizar pruebas de usabilidad, ya que son, por naturaleza, estáticos. Cuando estos bocetos son, además, utilizados en las pruebas de usabilidad del sistema con usuarios reales y expertos en usabilidad, la técnica se denomina Prototipado Interactivo en Papel.

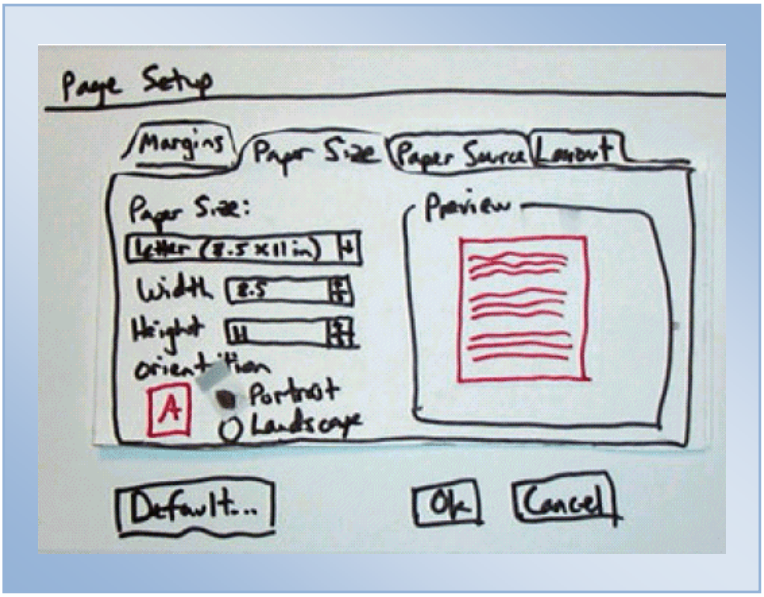


Figura 1.1: Esquema de una interfaz realizada en papel

1.3.5 Prototipado Interactivo en Papel

El prototipado Interactivo en Papel es un método empleado en pruebas de usabilidad que utiliza prototipos de baja fidelidad y suministros comunes de oficina (marcadores, transparencias, tijeras, etc.) para simular el comportamiento de un software. Los prototipos utilizados son a menudo bosquejos trazados a mano, aunque pueden utilizarse también capturas de pantallas de interfaces reales. El propósito de esta clase de prototipo es proveer un sentido de la dinámica de un programa sin tener en realidad una versión funcional del mismo.

1.3.6 Funcionamiento

El test de usabilidad se desarrolla en conjunto con uno o dos

desarrolladores que juegan el rol de computadora, manipulando las piezas de papel para simular el comportamiento del sistema. Los usuarios participantes en el test tienen como objetivo realizar tareas reales (asignadas con anticipación) interactuando directamente con el prototipo "*dando clics*" al tocar los botones o links e "*ingresando datos*" al escribir en las cajas de texto del prototipo presentado. En la figura 1.2 puede observarse la fotografía de una sesión donde se lleva a cabo un test de este tipo.

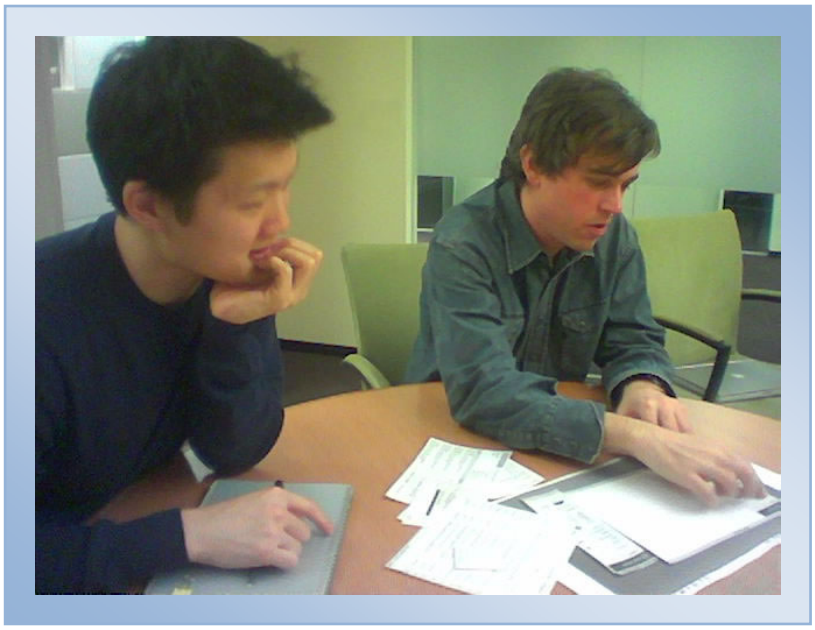


Figura 1.2: Sesión de Prototipado Interactivo en Papel

Un facilitador (usualmente alguien entrenado en usabilidad) conduce la sesión mientras otros miembros del equipo de desarrollo observan y toman apuntes. El "*computador*" no explica cómo se supone que debe trabajar la interfaz sino que se limita a simular lo que la interfaz

haría. De esta manera, es posible identificar qué partes de la misma son auto-explicativas y cuáles son confusas. Dado que los prototipos utilizados son en papel, es posible modificarlos fácilmente para solucionar los problemas encontrados.

El prototipado interactivo en papel es especialmente útil para recolectar información acerca de la siguiente clase de problemas:

- Conceptos y Terminología.- Ayuda a conocer si los usuarios realmente entienden los términos escogidos para la interfaz.
- Navegación y Flujo.- Cuando se trata de procesos o secuencias de pasos, es posible verificar si estos se realizan en la forma en que los usuarios esperan y si es evidente la necesidad de mecanismos para avanzar y/o retroceder.
- Contenido.- Ayuda a conocer si la interfaz provee la información correcta para las decisiones que los usuarios deben tomar y si existe información innecesaria o inapropiada.
- Layout.- A pesar de que las interfaces bosquejadas pueden no ser perfectas en apariencia, es posible evaluar si los usuarios pueden o no encontrar la información que necesitan para realizar sus tareas, si el orden de los campos de entrada es el adecuado y si la cantidad de información es la correcta, excesiva o insuficiente.

- Funcionalidad.- Es posible conocer ciertas funcionalidades que los usuarios necesitan o que han sido planificadas, pero que no son aprovechadas.

Por otra parte, esta clase de prototipado no es aconsejable cuando se desea conocer detalles acerca de:

- Factibilidad Técnica.- Los prototipos en papel no demuestran capacidad técnica. Es posible crear prototipos de papel que no pueden ser implementados en la realidad. Para evitar esto, es aconsejable incluir en el equipo de prototipado, al menos a una persona que conozca las limitaciones técnicas que involucra el proyecto.
- Tiempo de respuesta.- Dado que es una persona quien simula el comportamiento del computador, el tiempo de respuesta que percibe el usuario durante las pruebas de usabilidad es artificial.
- Colores y Fuentes.- Si realmente se necesita mostrar cómo lucirá el prototipo construido en la pantalla de un computador, los prototipos en papel no sirven. Por ello, es importante incluir en el ciclo de prototipado a diseñadores que puedan encontrar aspectos que sugieran la apariencia visual de la interfaz final.

1.4 Problema y Justificación

En el desarrollo de un sistema, es normal que las interfaces de usuario experimenten cambios sustanciales entre el concepto inicial y la versión final. Mediante la creación de prototipos es posible evitar correcciones excesivas e innecesarias del código escrito ya que constituye un excelente mecanismo para encontrar problemas significativos en las etapas iniciales del proceso de desarrollo y permite experimentar de forma iterativa hasta encontrar la mejor solución antes de invertir esfuerzo en implementarla.

Una ventaja adicional de crear prototipos es que éstos modelan el dominio específico de la aplicación sin considerar los aspectos técnicos en gran detalle. Su objetivo principal es hacer de una solución una propuesta más comunicable [17].

En años recientes, el desarrollo de herramientas CASE para soporte del proceso de desarrollo de software ha experimentado un incremento gradual. Las actuales versiones de Entornos de Desarrollo Integrado (IDEs por sus siglas en Inglés) como Netbeans [4], Visual Studio [5] o Eclipse [13] proveen opciones que permiten al desarrollador construir interfaces gráficas sin necesidad de escribir grandes porciones de código fuente. Estas herramientas de alto nivel permiten a los programadores, mediante sencillas operaciones de arrastrar y soltar, especificar qué se debe hacer, en lugar de cómo

debe hacerse, lo cual reduce ampliamente el tiempo requerido al momento de diseñar una interfaz de usuario.

Sin embargo, a pesar de que estas herramientas incluyen procedimientos que permiten especificar las opciones y tareas fundamentales utilizadas en la mayoría de las GUIs, su utilidad es evidente de forma exclusiva a personas con experiencia previa en el desarrollo de aplicaciones y conocimientos de programación, es decir, miembros de un equipo de desarrollo.

Otro inconveniente de estas herramientas es que a menudo fuerzan a especificar más detalles de los que se dispone y se necesita en la etapa de levantamiento de requerimientos [1], lo cual hace que sean con frecuencia, más que un beneficio, un inconveniente para el equipo del proyecto en las primeras etapas del desarrollo de software, donde una de las tareas principales consiste en crear prototipos no funcionales y de baja fidelidad que sirvan a lo largo del proceso de identificación de las necesidades del cliente, usuarios finales y demás involucrados en el proyecto.

La mayoría de las organizaciones de desarrollo de software construyen a menudo prototipos para evaluar diferentes alternativas de diseño y obtener así retroalimentación de parte de los usuarios en una forma práctica [3]. Sin embargo, los usuarios finales no son frecuentemente integrados a los ciclos de prototipado (mientras se

elaboran las distintas versiones de una interfaz), sino más bien consultados una vez que una visión coherente del sistema como un todo ha sido construida [1].

La participación de los interesados en la construcción de un sistema debería ser, por tanto, un factor a considerar durante las sesiones en las que los prototipos son elaborados por los diseñadores de interfaces. Esto haría posible que sus expectativas queden plasmadas sobre los prototipos en el mismo instante en que éstos son creados sin tener que esperar hasta las presentaciones planificadas por el equipo de desarrollo.

En vista de las limitaciones para interactuar con el cliente que presentan estas herramientas para creación de interfaces gráficas, muchos ingenieros de requerimientos prefieren –en lugar de utilizar software especializado– construir sus diseños como simples bosquejos o esquemas a mano alzada sobre papel que sugieran la apariencia de las interfaces que implementarán las funcionalidades solicitadas para el sistema. Una razón adicional para emplear este mecanismo es que estos productos no ofrecen las condiciones necesarias para lograr una coordinación efectiva entre las ideas de los clientes y la de los ingenieros de requerimientos.

Este problema de comunicación obedece mayormente a las diferencias inherentes a las correspondientes áreas de trabajo de las

partes involucradas y a los intereses que cada uno persigue mientras un sistema está siendo desarrollado. Así, por una parte, los ingenieros y desarrolladores, suelen ver al prototipado como un proceso en el que no tienen lugar mayores avances de implementación, dado que tienen la percepción de que la escritura de código fuente es una parte esencial de su trabajo. Mientras que en otra perspectiva, los interesados en el sistema tienden a esperar que los prototipos ejecuten todas las funcionalidades que aparentan, lo cual, en las etapas iniciales del desarrollo de un proyecto es virtualmente imposible.

Por esto último, resulta evidente la necesidad de la existencia de aplicaciones que puedan ser usadas a lo largo de la etapa de levantamiento de requerimientos, específicamente durante la creación de los prototipos de las interfaces de usuario que se diseñan para un sistema particular. Aplicaciones que permitan unificar las ideas generadas en una reunión entre clientes y usuarios finales y el equipo de desarrollo de software y representarlas en un prototipo de baja fidelidad que pueda ser usado en las evaluaciones posteriores del sistema en construcción.

El presente proyecto de tesis pretende construir una solución que pueda ser utilizada como soporte en la creación de prototipos no funcionales y de baja fidelidad durante la etapa de levantamiento de

requerimientos en el desarrollo de un proyecto de software.

La solución propuesta en el presente trabajo es PenGUIn Prototyper (Pen-based GUI Prototyper), una herramienta de prototipado cuyo alcance y las funcionalidades que pretende implementar se describen en secciones posteriores de esta tesis.

Se busca que esta herramienta permita a los diseñadores bosquejar una interfaz de manera rápida y sencilla, mostrar el flujo de navegación entre las interfaces bosquejadas y simular la ejecución de algunos de los elementos contenidos en estos diseños.

CAPÍTULO II.

2 MARCO TEÓRICO

Este capítulo detalla la revisión de varias de las tecnologías actuales que atacan los problemas relacionados a la creación de prototipos a mano alzada. Los paradigmas de interacción, el reconocimiento de trazos a mano alzada y algunas de las soluciones existentes para la creación de prototipos de baja fidelidad son detallados a fin de plantear una visión general del marco teórico sobre el cual se trabajará en las secciones posteriores.

2.1 Paradigmas de Interacción

Las interfaces de usuario son el medio de interacción entre los usuarios y el sistema. Al diseñar una interfaz, una de las primeras decisiones que se deben tomar es la de qué paradigma de interacción se aplicará en el desarrollo. De manera general, un paradigma es el modelo, filosofía o principios bajo los cuales se diseña una interfaz.

A menudo, el usuario se encuentra limitado a un paradigma de interacción específico en la implementación de un sistema de software. En la actualidad, se observa la convergencia de varios

factores:

- Ubicuidad de ciertos dispositivos (punteros y mouse)
- Alto costo de dispositivos alternativos (ej. Superficies táctiles, cámaras de video)

Estos dos factores, de cierta manera, han derivado en la *estandarización* de un paradigma de interacción en particular: El Paradigma del Escritorio.

2.1.1 El Paradigma del Escritorio

El paradigma del escritorio ha sido la metáfora predominante durante muchos años desde su introducción en el Xerox PARC. La primera computadora personal en popularizar este tipo de interfaces fue la Apple Macintosh en 1984.

El escritorio dispone de ventanas, iconos, menús y punteros. Este estilo utiliza un mecanismo físico (el mouse) para controlar la posición de un cursor en pantalla. La información se muestra en ventanas, agrupada en menús y es mayormente representada por íconos.

Las ventajas inherentes a este paradigma se centran en la facilidad de uso, debido a la baja carga cognitiva que requiere el acceder a las distintas acciones disponibles. El otro factor importante, es la unificación de la apariencia de la interfaz, debido a esto, la curva de

aprendizaje al pasar de un programa a otro es sumamente baja.

De forma paralela a la constante evolución del paradigma del escritorio han surgido nuevos paradigmas de interacción, cuya aparición sugiere mejoras en la forma y facilidad de uso de los sistemas.

2.1.2 Computación Ubicua

Mark Weiser, describió este paradigma post-escritorio, como la tercera era de la computación:

- La primera dominada por el esquema cliente servidor, donde muchos comparten la tecnología, a través de grandes máquinas o “mainframes”.
- La era de los computadores personales. Cada usuario posee un computador personal, donde almacena su información e interactúa directamente con ella.
- La tercera era, de la computación ubicua, donde muchas computadoras, comparten la interacción con un mismo ser humano.

Para llegar a este objetivo, identificó que la tecnología debía volverse *transparente*. Al utilizar una PC, nos hemos acostumbrado a concentrarnos en la interacción con la misma. Al contrario, si un grupo grande de computadoras, deben de compartir nuestro tiempo

para interactuar con nosotros, éstas deben ser lo menos intrusivas posible.

Tecnología Involucrada

El periodo de transición hacia computación ubicua está caracterizado por la evolución de varias tecnologías claves para su aplicación. En la actualidad estas tecnologías se encuentran inmiscuidas en nuestra actual red global, el internet [14].

El protocolo actual de direccionamiento, IPv4, ya se encuentra en sus límites, con respecto a la cantidad de objetos para direccionar. La siguiente versión del protocolo IPv6, con direcciones de 128 bits, sobrepasará con gran margen este límite, lo cual nos permitiría poner en práctica el objetivo de direccionar todos pequeños computadores con que trabajaremos en el mundo de la computación ubicua.

- Computación Distribuida: Los protocolos actuales de comunicaciones, fueron diseñados principalmente con el esquema cliente-servidor en mente. Por lo tanto, las comunicaciones, tolerancia a fallos y otros factores no están ajustados para ambientes de computación altamente distribuida. En la computación ubicua tratamos con cientos de computadores compartiendo la tarea de atender a un usuario, los métodos de comunicación deberán de ser más dinámicos y

flexibles.

- Redes Inalámbricas: Las redes necesarias para comunicar a dispositivos ubicuos, fácilmente sobrepasarán los estándares de las redes de computadoras convencionales. Tanto en ancho de banda como en flexibilidad. Las redes actuales no están diseñadas para manejar una alta densidad de dispositivos. Tampoco facilitan el tránsito de los dispositivos desde una red a otra, impidiendo su movilidad.

2.1.3 Bits Tangibles

Las interfaces de bits tangibles (TUIs por sus siglas en inglés), proveen forma física a la información digital y a la plataforma de computación. El objetivo principal de dichas interfaces es facilitar la manipulación directa de la información.

Como seres humanos, desde pequeños aprendemos y ganamos destreza en la manipulación de objetos en el mundo físico. Sin embargo, con la predominancia de las GUIs, estas habilidades no son aprovechadas al momento de interactuar con la información digital.

Ishii [15] sugiere la imagen del ábaco, como un ejemplo ideal de las TUIs. De manera sencilla de operar, las piezas del ábaco son, al mismo tiempo, los mecanismos de entrada y salida de la información.

Ambientes Inteligentes

Otro de los enfoques de la investigación en bits tangibles, consiste en aprovechar la información ambiental que los usuarios reciben a través de sus sentidos periféricos. Por ejemplo:

- En su oficina, un usuario consulta constantemente –sin desenfocar su atención– el estado del clima si dispone de una ventana cercana.
- Caminando por la calle, un grupo de turistas determina la hora aproximada al escuchar el campanario de una iglesia cercana.
- El “Live Wire” de Xerox PARC, es una de las demostraciones más famosas de cómo mezclar información digital en el medio ambiente, de una manera poco obstrusiva.

2.2 Reconocimiento de Trazos a Mano Alzada

El reconocimiento de trazos a mano alzada comprende el reconocimiento automático de trazos dibujados a mano, mediante el uso de dispositivos táctiles. Es un área de estudio correspondiente a la inteligencia artificial y el reconocimiento de patrones.

La tecnología desarrollada es de especial interés en el área de Interacción Hombre-Máquina ya que es posible reemplazar las interfaces convencionales, por otras más naturales que tomen

ventaja de la habilidad, casi instintiva, que poseemos para dibujar.

En la actualidad ha tomado fuerza su aplicación en dispositivos móviles, debido a que se logra ahorrar espacio al integrar los módulos de entrada y salida mediante la sustitución de pantalla y botones por una superficie táctil. Todos estos factores han generado un creciente interés en este campo de investigación.

Los algoritmos de reconocimiento de trazos, han divergido en dos acercamientos distintos [16]:

- Por Reconocimiento de Gestos.- Los algoritmos de este tipo suelen tener un alto grado de efectividad, siendo ésta su mayor virtud. Funcionan identificando factores como: número de trazos, orden, velocidad y dirección. La principal limitación de estos algoritmos radica en que el usuario debe pasar por un proceso de aprendizaje/adaptación de los gestos válidos en un sistema.
- Por Visión.- Están basados en el reconocimiento de características o propiedades geométricas. Aunque la efectividad es modesta, los usuarios no necesitan pasar por el proceso de aprendizaje y pueden dibujar sin mayores restricciones.

La mayoría de sistemas implementados en la actualidad, se encuentran en un punto intermedio entre los dos enfoques

mencionados. Se centran en dominios específicos, limitando al usuario a un conjunto de figuras predeterminadas.

2.2.1 Sistemas Multi-dominio para el Reconocimiento de Trazos

Los actuales sistemas que utilizan reconocimiento de trazos en su interfaz son desarrollados casi exclusivamente por investigadores del área. Los algoritmos actuales, con su grado de efectividad, son enfocados a dominios específicos, lo que obliga al desarrollador a modificar o reimplementar el algoritmo por cada dominio de trabajo fijado. La mayoría de algoritmos utilizan también técnicas de "machine learning" para entrenar al sistema en el reconocimiento de las figuras válidas en un dominio particular.

Desarrollar sistemas con este esquema es extremadamente tedioso. Los resultados en el reconocimiento son dependientes del desarrollador y del entrenamiento previo realizado [16].

Una de las nuevas propuestas en el área de reconocimiento de trazos, es diseñar reconocedores genéricos a fin de construir sistemas de reconocimiento multidominio. LADDER, una de estas propuestas, es un framework de reconocimiento de trazos multidominio desarrollado en el "Sketch Recognition Lab" de Texas A&M que se enfoca en el reconocimiento de figuras primitivas que juntas pueden dar significado a figuras más complejas o de más alto

nivel.

Al programar una interfaz basada en LADDER, se describen las relaciones necesarias entre figuras primitivas para obtener figuras más complejas, lo que hace posible representar figuras de un dominio particular.

El *contexto* es otro componente importante en el reconocimiento de trazos aplicado a dominios específicos. En casos donde geométricamente existe la misma incertidumbre al tratar con una figura u otra, el contexto puede ayudar a enmarcar mejor el problema. Por ejemplo, si el dominio estudiado es la diagramación de redes eléctricas, una poli-línea en zigzag, interpretada como un tachón para borrar en otros dominios, sería reconocida de forma correcta como una resistencia.

2.2.2 Reconocimiento de Escritura

El reconocimiento de escritura es la capacidad de un ordenador de recibir una entrada manuscrita e interpretarla. Dependiendo del tipo de dispositivo utilizado es posible hablar de reconocimiento "en línea" (cuando se trata de superficies táctiles) o reconocimiento "fuera de línea" (por ejemplo, al escanear un documento impreso).

Al igual que en el reconocimiento de trazos, algunos sistemas imponen restricciones sobre la forma de escribir los caracteres con el

propósito de mejorar la efectividad del reconocimiento realizado:

- Un sistema fuera de línea podría presentar problemas para reconocer los diferentes tipos de escritura de cada persona. Sin embargo, si se limita la escritura a un conjunto formado por caracteres específicos, se facilita en gran manera el trabajo que debe realizarse para el reconocimiento. Esto podría ser aplicado, por ejemplo, al extraer números telefónicos (o de alguna otra índole) de un formulario estructurado.
- El sistema Graffiti de Palm, plantea un sistema de escritura basado en gestos. En este sistema, los usuarios deben aprender un conjunto de símbolos que contiene un gesto por cada símbolo del alfabeto y demás caracteres alfanuméricos. La efectividad de este sistema es cercana al 100% con un usuario experto.
- El sistema de reconocimiento propuesto por Microsoft Ink utiliza información de contexto y del lenguaje. Si algunos caracteres no logran ser reconocidos con suficiente certeza, el sistema puede consultar un diccionario del lenguaje para determinar cuáles son las opciones más probables. Este acercamiento, sin embargo, dificulta la escritura en varios idiomas a la vez.

2.3 Algunas Soluciones Existentes

Empleando las tecnologías planteadas en la sección anterior, varias estrategias han sido propuestas para superar las limitaciones de las herramientas tradicionales para la creación de prototipos tempranos. Con el objetivo de promover una interacción más natural con el usuario, se han diseñado soluciones que utilizan paradigmas de interacción diferentes a los utilizados habitualmente en ambientes de computación tradicional, lo que ha generado la aparición de unas pocas herramientas que implementan el soporte de trazos a mano alzada durante la construcción de interfaces gráficas de usuario. Las principales características de dos de estas soluciones existentes son detalladas a continuación con el propósito de ilustrar las particularidades de una herramienta de esta naturaleza.

En secciones posteriores de esta tesis se explicarán en detalle las opciones soportadas por PenGUIn Prototyper y las que consideramos ventajas sobre las herramientas existentes. Cabe mencionar que las soluciones presentadas a continuación son prototipos creados a nivel académico o de investigación pues no existen hasta ahora productos en el ámbito comercial que ataquen el problema de la creación de prototipos de interfaces gráficas de usuario utilizando trazos a mano alzada.

2.3.1 SILK

SILK [21, 22] (Sktech Interfaces Like Krazy) es una herramienta para prototipado de interfaces gráficas que permite al usuario bosquejar interfaces de manera rápida utilizando dispositivos de entrada electrónicos como tablas digitalizadoras y stylus. SILK, que reconoce ventanas y otros elementos de una interfaz a medida que son dibujados por el usuario, preserva las características más importantes del lápiz y el papel ya que permite construir bosquejos rápidamente en una superficie bastante flexible. Sin embargo, a diferencia del papel, los trazos construidos sobre medios electrónicos generan un ambiente interactivo y pueden ser fácilmente modificados.

SILK implementa además mecanismos que guardan la historia de los bosquejos de una interfaz, lo que hace posible reutilizar porciones de diseños anteriores en versiones más recientes de una misma interfaz para propósitos de pruebas o comparaciones. Esta última característica permite crear interfaces que pueden evolucionar a través del tiempo, sin forzar al diseñador a recomenzar sus diseños cada vez que estos son requeridos.

SILK permite transformar los bosquejos construidos a mano alzada en interfaces reales con una apariencia específica. La figura 2.1 muestra la interfaz transformada de un bosquejo construido a mano

alzada. En esta figura es posible notar además que SILK puede reconocer secuencias horizontales o verticales de elementos de una interfaz y mostrarlos alineados en la interfaz real.

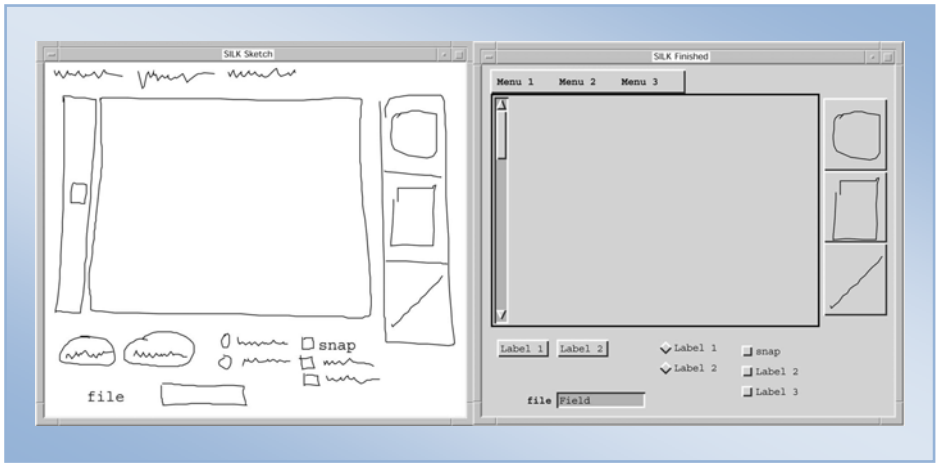


Figura 2.1: Bosquejo e Interfaz de un diseño construido en SILK

Forma de *Interacción*

Como se observa en la figura 2.2, que muestra parte de la interfaz de SILK, ésta se compone de elementos de interfaces tradicionales (ventanas, menús, barras de desplazamiento, etc.). En la ventana de la parte inferior derecha se observa el bosquejo de una interfaz para información del clima en la que el diseñador ha bosquejado el pronóstico del tiempo para cinco días. En las figuras del fondo, puede observarse el modo de Storyboard de SILK. En esta parte, el usuario especifica el comportamiento de la interfaz mediante líneas que parten de los diferentes elementos de la misma indicando el flujo de

navegación entre las ventanas dibujadas.

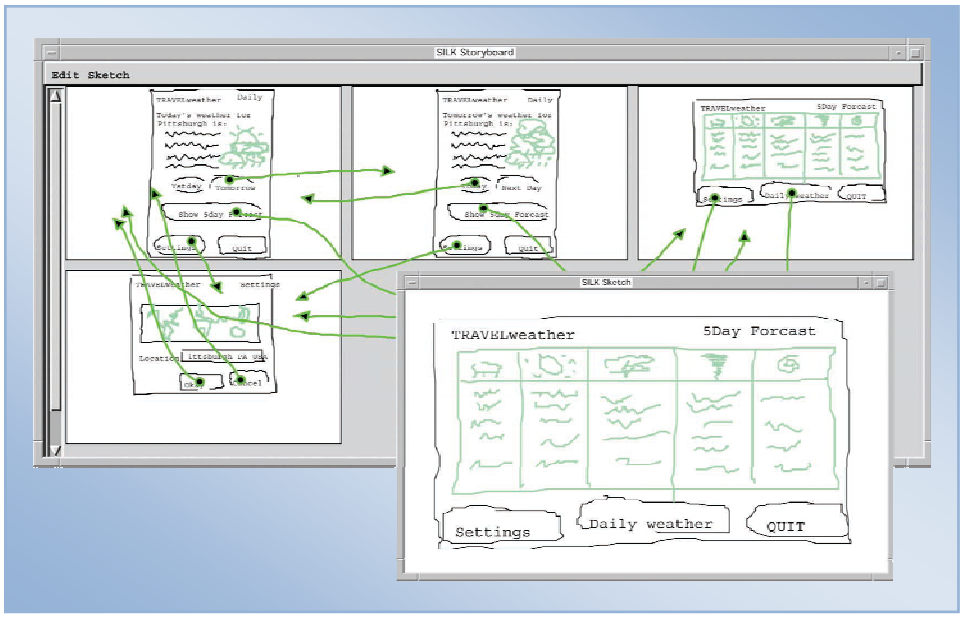


Figura 2.2: La Interfaz de SILK

SILK permite la interacción con el usuario mediante gestos para editar y redibujar porciones de los bosquejos trazados. La figura 2.3 muestra, ordenados de izquierda a derecha, los gestos para eliminar, agrupar, desagrupar y convertir un elemento de la interfaz en la mejor inferencia hecha por SILK. El último gesto mostrado permite al usuario insertar texto desde el teclado en el diseño o reemplazar un garabato que representa texto. Las flechas amarillas indican la dirección en la que deben dibujarse los gestos para lograr un mejor reconocimiento.

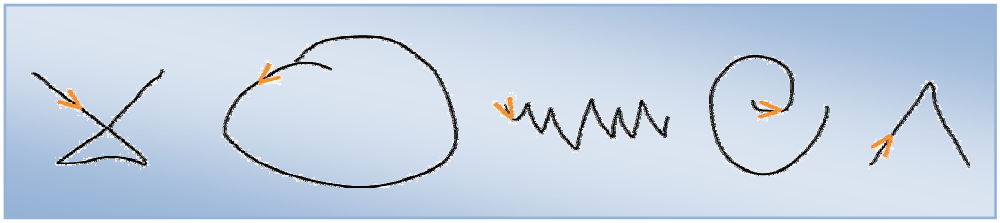


Figura 2.3: Gestos para edición soportados por SILK

2.3.2 DENIM

DENIM (Design Environment for Navigation and Information Models) [19, 20], un proyecto del departamento de Interacción Hombre-Máquina de la Universidad de Washington, es un sistema que ayuda a diseñadores de sitios web a ilustrar la navegación e interacción de un sitio en las etapas iniciales del diseño. Permite esquematizar la estructura de las páginas web que componen el sitio mediante trazos informales, crear links entre éstas e interactuar con los prototipos bosquejados en un modo de ejecución.

Cinco niveles diferentes de edición son puestos a disposición del usuario con el fin de que las interfaces construidas puedan ser diseñadas con mayor o menor detalle, dependiendo de la fidelidad con la que se las desee representar. Estos niveles de edición pueden ser accedidos mediante el uso de un zoom semántico que adapta la vista actual de la interfaz al contexto correspondiente en función del nivel de acercamiento/alejamiento especificado por el usuario mediante una barra de desplazamiento provista en la interfaz o a

través del uso del mouse.

Los niveles de edición del sitio en los que es posible bosquejar los diseños son:

- Overview.- Provee un nivel bastante alto de abstracción que provee una vista general del sitio completo.
- Site Map.- Muestra una vista del sitio representando a las páginas que lo componen como etiquetas que contienen miniaturas de las páginas y están conectadas entre sí que ilustran los vínculos entre ellas.
- Storyboard.- Este nivel de edición permite al usuario ver de forma simultánea varias páginas y las relaciones de navegación entre ellas con un mayor grado de detalle.
- Page.- Muestra el 100% del tamaño de una página y permite a los usuarios especificar con bastante detalle los contenidos de la misma.
- Detail.- Provee un nivel de edición mucho más fino para precisar detalles de los componentes de las páginas con mayor precisión.

La figura 2.4 muestra la interfaz de DENIM en el nivel de edición de Storyboard donde se ha bosquejado un sitio web compuesto por nueve páginas. La forma en la que estas páginas se vinculan es

representada con las líneas verdes trazadas por el usuario desde los componentes que activan el flujo de navegación entre ellas, lo que además ilustra el comportamiento que tendrá el sitio una vez finalizado.

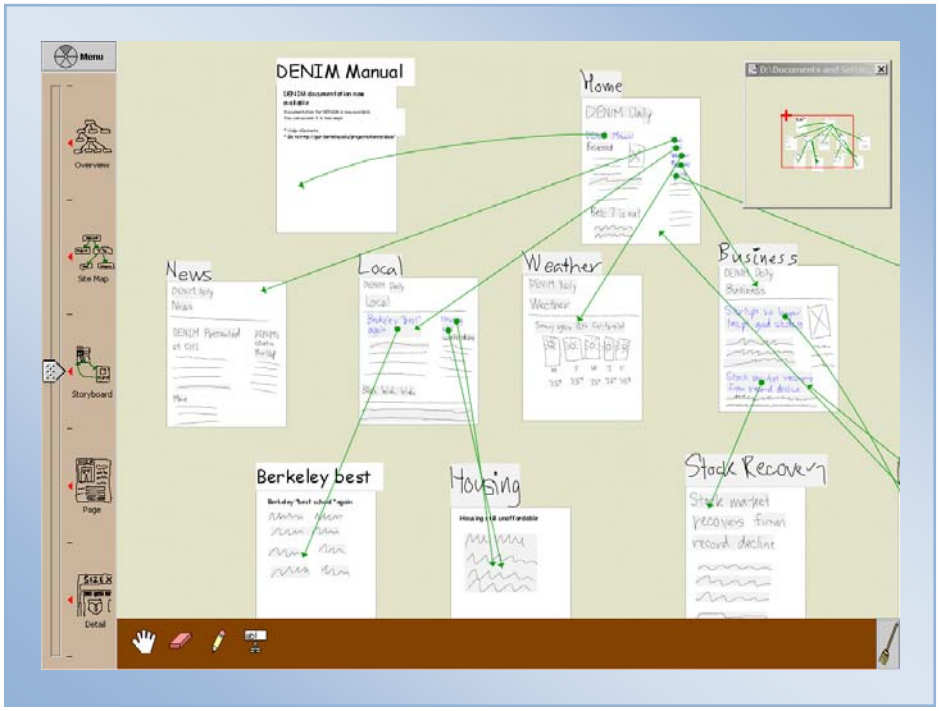


Figura 2.4: Modo Storyboard de DENIM

Forma de Interacción

La interacción propuesta por DENIM es una interacción no tradicional. Además de explotar el uso de esquemas trazados a mano alzada, utiliza un conjunto relativamente pequeño de gestos para realizar determinadas tareas en la interfaz: las que tienen que ver con el desplazamiento del lienzo sobre el cual son dibujados los

elementos de la interfaz. En lugar de los menús tradicionales que despliegan submenús en forma vertical se proponen menús circulares en forma de pie como los que se muestran en la figura 2.5.

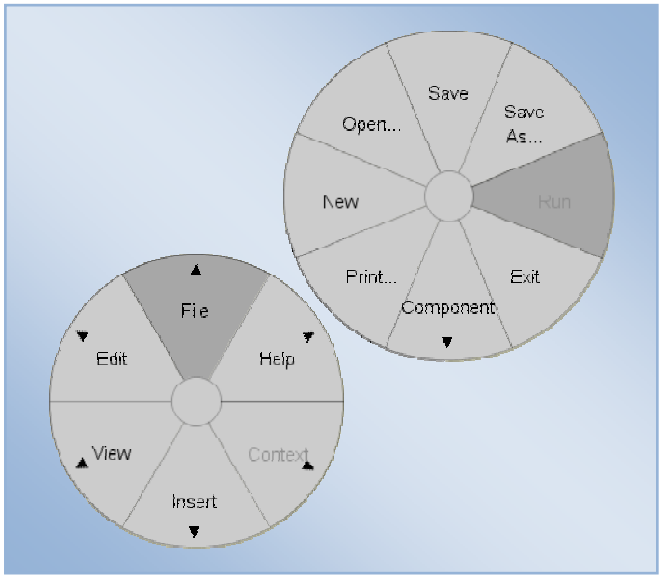


Figura 2.5: Menús en Forma de Pie

Características Adicionales

DENIM no efectúa ningún reconocimiento de los trazos realizados por el usuario, con excepción del conjunto de gestos descritos anteriormente y las líneas que se dibujan entre dos páginas para representar los links entre ellas. A diferencia de SILK, que intenta reconocer los trazos realizados y mostrar al usuario su interpretación lo más pronto posible a medida que el éste realiza los trazos, DENIM

evita, intencionalmente, efectuar esta clase de reconocimiento para mejorar la eficiencia y el tiempo de respuesta.

La versión 2.1 de DENIM, liberada en Abril del 2007, provee además la opción de ingresar texto mediante el uso del teclado usando una caja de texto para dicho propósito.

2.4 Solución Planteada: PenGUIn Prototyper

El presente proyecto de tesis pretende proponer una solución que pueda ser utilizada como soporte en la creación de prototipos no funcionales y de baja fidelidad durante la etapa de levantamiento de requerimientos en el desarrollo de un proyecto de software.

La solución propuesta en el presente trabajo es PenGUIn Prototyper (Pen-based GUI Prototyper), una herramienta de prototipado a mano alzada.

Se busca que esta herramienta permita a los diseñadores bosquejar una interfaz de manera rápida y sencilla utilizando la pantalla táctil de una Tablet PC combinando la versatilidad del papel y aprovechando el soporte provisto por estos dispositivos digitales para anotar ideas y trazar gráficos que pueden ser luego comunicados a otros colaboradores [18].

La herramienta construida no sólo permitiría trazar bosquejos de interfaces gráficas a mano alzada sino además editar y corregir

errores en las interfaces diseñadas, características no disponibles o no muy viables al utilizar una pieza de papel.

Con el propósito de que los prototipos construidos puedan ser utilizados para conocer y dimensionar las expectativas que el cliente y los usuarios finales tienen acerca del sistema, se espera que PenGUIn Prototyper sea además una herramienta que permita mostrar el flujo de navegación entre las interfaces bosquejadas y que permita simular la ejecución de algunos de los elementos contenidos en estos diseños.

CAPÍTULO III.

3 ANÁLISIS Y DISEÑO

Este capítulo describe el Análisis y Diseño del sistema. Se exponen las diferentes decisiones de diseño tomadas como resultado del análisis y se puntualizan los casos de uso y el detalle de los módulos y componentes del sistema. El diseño lógico, de interacción con el usuario y las pruebas que se realizarán más adelante son también revisados en este capítulo.

3.1 Análisis del Sistema

La solución descrita en este documento es una herramienta que permite al usuario bosquejar a mano alzada prototipos no funcionales y de baja fidelidad de interfaces gráficas de usuario. Debido a que el reconocimiento de trazos es un área de investigación relativamente nueva, se han definido algunas limitaciones con respecto a las funcionalidades a ser implementadas en el sistema.

Requisitos Funcionales

- Reconocer los trazos dibujados por un usuario, en el contexto del prototipado de Interfaces Gráficas.
- Exportar las interfaces construidas por el usuario, con el

propósito de que éstas puedan ser utilizadas en otros programas especializados en la creación de interfaces gráficas.

- Permitir la visualización de cómo lucirían los diagramas trazados en interfaces gráficas reales. Y permitir la interacción del usuario con las interfaces construidas.

Requisitos no Funcionales

- La interacción con el sistema debe ser orientada al uso de superficies táctiles.
- El sistema debe maximizar la facilidad de uso y tolerancia. Un usuario inexperto, deberá poder utilizar el sistema con poca instrucción y sentirse cómodo al realizar las tareas permitidas en el mismo.

3.2 Casos de Uso

El único rol en el sistema es el de Usuario. Al captar la entrada de los trazos realizados por el usuario, el sistema interpreta el mismo tipo de evento sin importar el medio de interacción disponible (pluma digitalizadora, pantalla táctil, mouse, etc.). En el caso de que varios usuarios trabajen de manera conjunta en un diseño, no se pretende distinguir entre cliente y diseñador u otra relación.

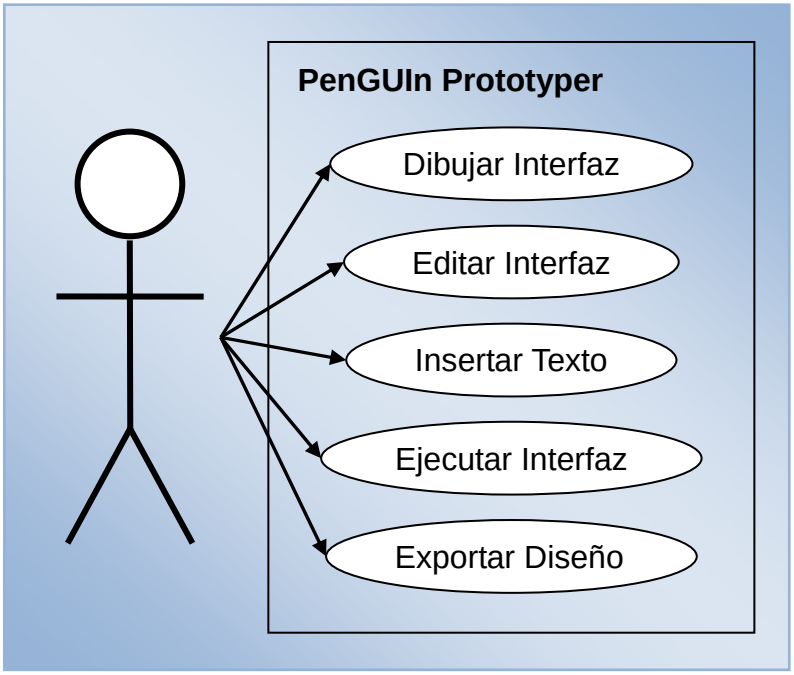


Figura 3.1: Casos de Uso del Sistema

La figura 3.1 es el diagrama de casos de uso del sistema. Los mismos que cumplen con los requisitos funcionales descritos en la sección anterior.

- Dibujar Interfaz.- El Usuario realiza los trazos a mano alzada sobre un lienzo vacío provisto por el sistema. A medida que los bosquejos son creados, el sistema provee retroalimentación acerca de la interpretación de los trazos.
- Editar Interfaz.- Mediante gestos, el Usuario puede editar los

trazos hechos sobre el lienzo. Al realizar un garabato sobre uno o más trazos, éste se elimina tachados son eliminados.

- Insertar Texto.- El sistema provee un teclado en pantalla, mostrado en la figura 3.2, para permitir el ingreso de caracteres. Debido a que la aplicación está pensada para una interacción puramente táctil, no se supone la presencia de un teclado físico.

Figura 3.2: Teclado en pantalla para ingreso de texto

- Ejecutar Interfaz.- Mientras el usuario bosqueja la interfaz, es posible visualizar los diseños en la forma en que éstos han sido interpretados por el motor de reconocimiento. Las interfaces producidas tienen el mismo nivel de detalle y permiten probar la interacción definida en los bosquejos.
- Exportar Diseño.- Para facilitar la continuación del trabajo, el

sistema provee la facilidad de exportar los resultados de la sesión. El archivo exportado es interpretable por un editor de interfaces graficas de diseño tradicional.

3.3 Diseño Lógico del Sistema

El único rol en el sistema es el de Usuario. Al captar la entrada de los trazos realizados por el usuario, el sistema interpreta el mismo tipo de evento sin importar el medio de interacción disponible (pluma digitalizadora, pantalla táctil, mouse, etc.). En el caso de que varios usuarios trabajen de manera conjunta en un diseño, no se pretende distinguir entre cliente y diseñador u otra relación.

El diseño lógico del sistema aprovecha las características de diseño sobre las que se basa GUILD, una herramienta propuesta como parte de la implementación de LADDER [24]. Este sistema de reconocimiento multi-dominio, toma como entrada los diferentes archivos descriptores de las figuras válidas en un dominio particular y los transforma en reconocedores de las figuras trazadas a mano alzada. Así, GUILD traduce los descriptores en instancias de objetos con capacidad para reconocer, mostrar y permitir la edición de las figuras trazadas por el usuario.

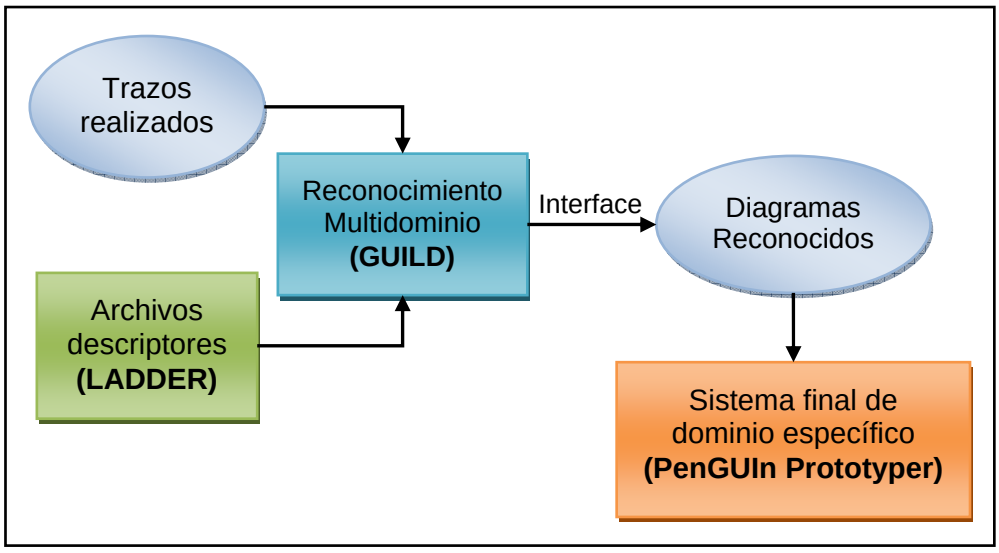


Figura 3.3: Diseño Lógico del Sistema

En la figura 3.3 se muestra el diseño lógico del sistema, el mismo que resulta de emplear la propuesta planteada en el trabajo de implementación de LADDER. La información del dominio específico sobre el cual se desea hacer el reconocimiento es detallada en los archivos descriptores de las figuras válidas en dicho dominio y esta descripción se provee a GUILD para evaluar las entradas del usuario en función de esta información y decidir así si los trazos realizados representan o no figuras válidas en este dominio.

Dado que el dominio particular del presente trabajo de tesis es el de las interfaces gráficas de usuario, el sistema construido deberá ser capaz de reconocer figuras que representen elementos de una interfaz gráfica, tales como ventanas, botones, listas desplegables, cajas de texto y demás componentes que podrían estar presentes en

una interfaz.

La descripción general las figuras que representarán a cada uno de estos elementos se especifica usando LADDER en los archivos descriptores del sistema. Estos archivos proveerán los criterios que el sistema evaluará para determinar si una figura es válida o no dentro del dominio de las interfaces de usuario. Las relaciones geométricas entre los elementos que componen estas figuras, la forma en la que deben ser mostradas una vez que han sido reconocidas y las diferentes características de edición disponibles para el usuario son los aspectos que se detallan en los archivos mencionados.

Una vez que el reconocimiento ha sido realizado, la información de las figuras reconocidas como válidas es enviada a PenGUIn Prototyper, el sistema de reconocimiento en un dominio específico (las interfaces de usuario), el mismo que implementa las funcionalidades específicas que se persiguen en esta tesis. Una interfaz Java permite la comunicación entre GUILD y la interfaz gráfica de PenGUIn Prototyper y coordina el paso de mensajes que son enviados desde GUILD hacia PenGUIn Prototyper y viceversa.

3.3.1 Los Archivos Descriptores

La figura 3.4 muestra la definición en LADDER de la figura que

representa una ventana en PenGUIn Prototyper. Según la definición planteada en el archivo mostrado, una ventana *es un* Contenedor *Abstracto* compuesto de dos figuras de más bajo nivel: Una de tipo *Rectángulo Abstracto* de nombre *rect* y una *Línea* llamada *bar*. Todos los tipos de figura usados en la definición de una ventana han sido definidos con anterioridad en otros archivos descriptores del sistema. Seguidamente en el archivo se han especificado las restricciones que deben cumplir los dos componentes (*rect* y *bar*) de una figura trazada para que ésta sea reconocida por el sistema como una ventana (la línea *bar* debe, por ejemplo, ser horizontal y estar contenida en el rectángulo *rect*). Por último, se indica que los componentes de cada ventana dibujada deben mostrarse en color negro, una vez que ésta ha sido reconocida por el sistema.

Archivo Window.lsd

```
(define shape Window
 (isA AbstractContainer)
 (components
    (AbstractRectangle rect)
    (Line bar)
 )
 (constraints
   (horizontal bar)
   (equalLength bar rect.boundTop)
   (contains rect bar)
   (near bar.center rect.boundTopMiddle)
 )
 (display
   (color black)
 )
)
```

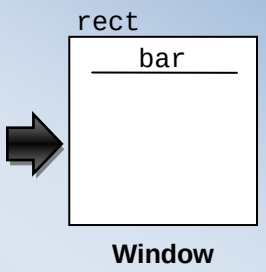


Figura 3.4: Descripción en LADDER de una ventana

LADDER soporta la definición jerárquica de descriptores. Esta característica del lenguaje hace que sea posible aprovechar las propiedades predefinidas en un descriptor haciendo que una nueva figura *extienda* del mismo. En PenGUIn Prototyper, por ejemplo, una ventana es, además, un *Contenedor Abstracto*, tipo de figura que ha sido definido en otro descriptor cuyas propiedades y características son tomadas como base para definir una ventana.

La figura 3.5 muestra el esquema de jerarquía formado por los archivos descriptores de las figuras válidas en el dominio tratado en PenGUIn Prototyper. En nuestro sistema, por ejemplo, un Cuadrado es un Rectángulo (hereda de Rectángulo) de lados iguales y éste, a su vez, es (hereda de) una Figura Primitiva.

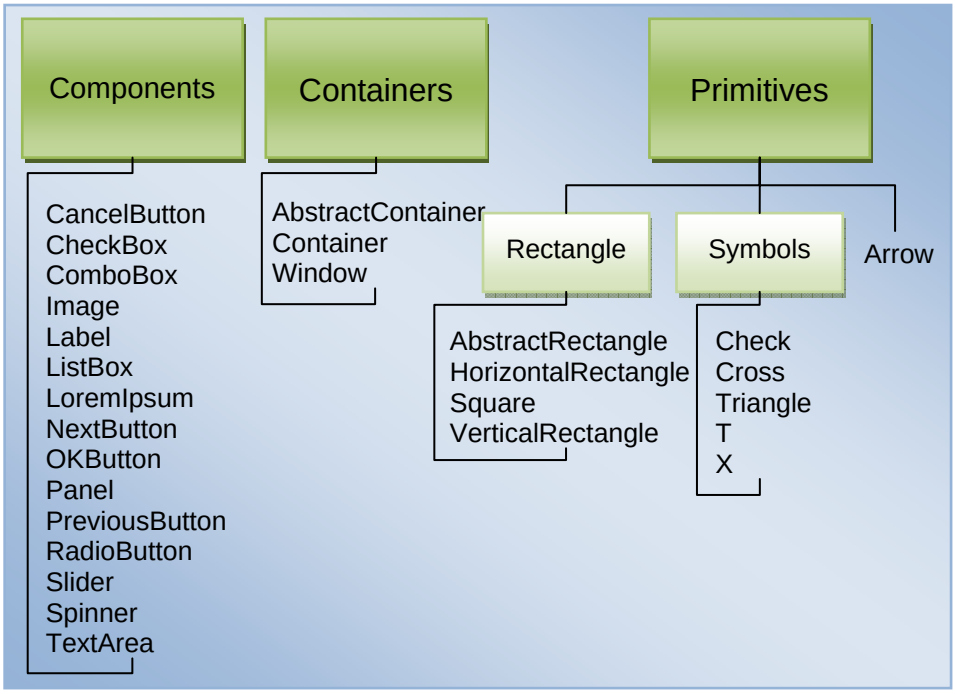


Figura 3.5: Esquema de Jerarquía de los descriptores del sistema

En este esquema de jerarquías, se listan además algunas figuras que no son propias del dominio de las interfaces gráficas de usuario. Estas figuras de *bajo nivel* han sido creadas con características que luego pueden ser aprovechadas por las demás a través de mecanismos como la herencia o la composición. Todas las figuras de alto nivel se componen de otras de más bajo nivel que no tienen significado por sí solas en el contexto del dominio de las interfaces de usuario. Se ha definido, por ejemplo, una figura denominada *T* que es utilizada dentro de un *Contenedor* para representar una caja de texto. Al momento del reconocimiento, cualquier trazo en forma de

T, que no esté dentro de un contenedor, es ignorado, pues no tiene sentido por sí solo dentro de este dominio.

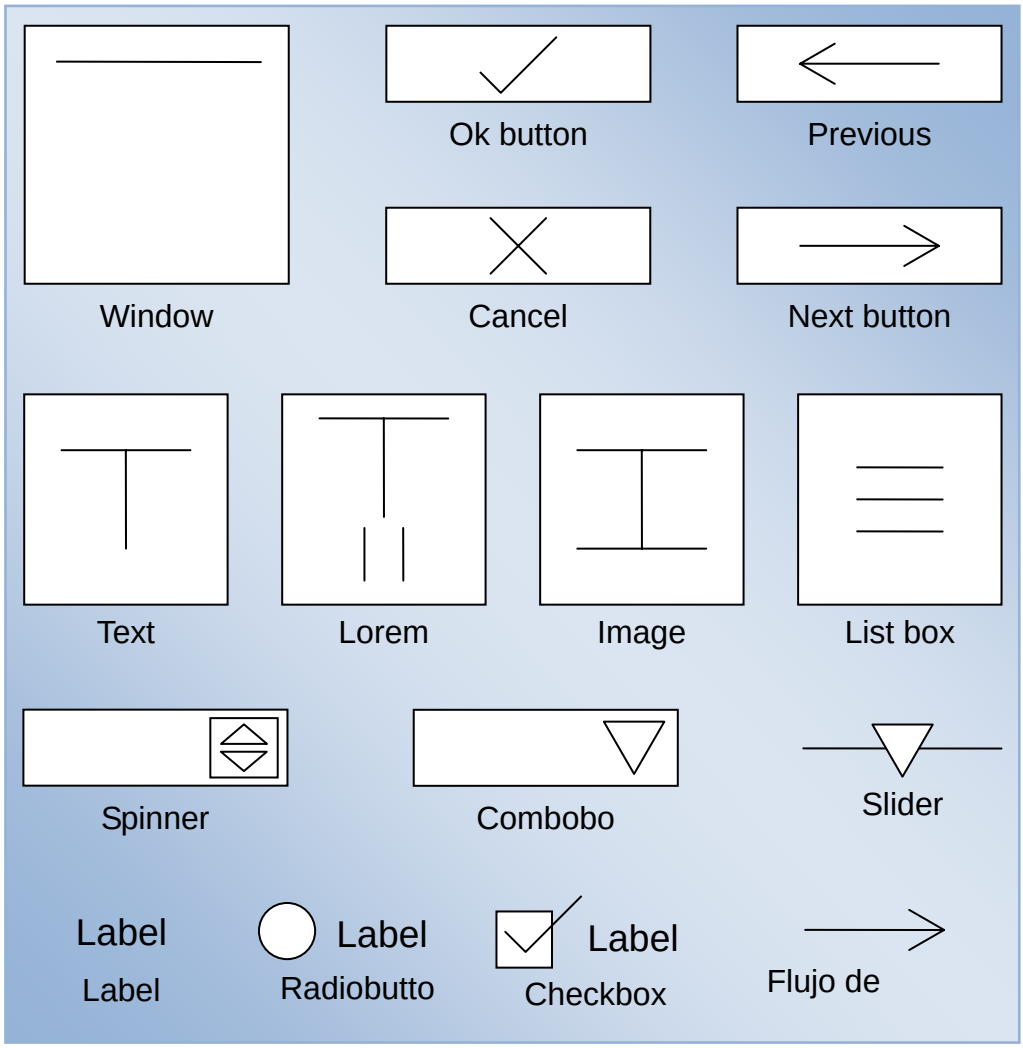


Figura 3.6: Figuras definidas en los archivos descriptores

En la figura 3.6 se muestran las figuras de alto nivel definidas en los archivos descriptores, es decir que son reconocidas como válidas en PenGUIn Prototyper, y su significado o lo que representan en el

dominio de las interfaces de usuario.

3.4 Detalle de los Módulos y Componentes del Sistema

PenGUIn Prototyper, como sistema de reconocimiento de trazos del dominio particular de las interfaces de usuario, se compone de tres módulos principales:

- La interfaz gráfica.- Provee las herramientas para la interacción del sistema con el usuario y constituye el canal de entrada de los trazos realizados.
- Generador de Componentes Swing/Java.- Es la parte del sistema que se encarga de interpretar los diagramas reconocidos por GUILD y traducirlos a componentes gráficos de Swing. Este módulo realiza dicha traducción instanciando objetos de la librería gráfica de java a medida que la Interfaz gráfica de PenGUIn Prototyper reporta las figuras de alto nivel, reconocidas en función de los archivos descriptores. Una vez que las interfaces Java han sido construidas, este módulo envía los resultados a la interfaz gráfica para ser luego mostrados al usuario en el modo de ejecución del sistema.
- Generador de Código QT Designer.- Este módulo interactúa con el Generador de Componentes Swing/Java, del que recibe como entrada las interfaces gráficas construidas para

representarlas en formato XML que pueda ser entendido por la herramienta de diseño de interfaces Qt Designer [23].

La salida generada por este módulo son archivos que contienen la representación XML de cada una de las ventanas bosquejadas por el usuario en la interfaz gráfica de PenGUIn Prototyper. Estos archivos pueden ser importados desde Qt Designer para una edición posterior con el soporte de todas las opciones provistas por esta herramienta.

La comunicación entre los diferentes módulos se realiza en forma directa mediante la invocación de métodos provistos por las interfaces definidas en los mismos que reciben los argumentos y retornan las salidas apropiadas para coordinar dicha comunicación.

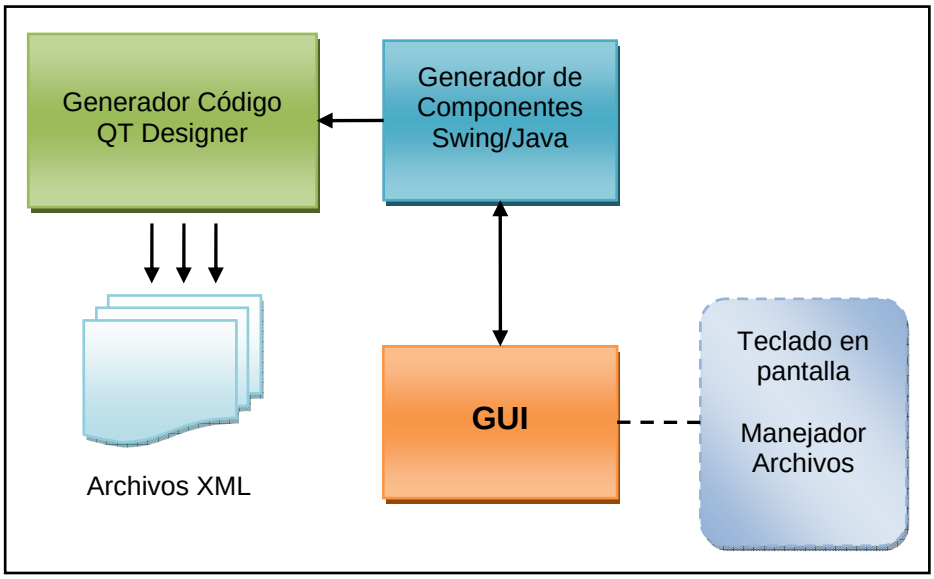


Figura 3.7: Esquema de Módulos del Sistema

La figura 3.7 muestra el esquema de los módulos del sistema y el flujo de comunicación entre éstos. Además de los tres módulos descritos anteriormente, se detallan dos herramientas utilizadas en la interfaz para soporte de algunas otras funcionalidades: el Teclado en Pantalla para ingreso de texto y un Manejador de archivos que tiene la función de administrar la exportación de los archivos creados por el Generador de Código de Qt Designer.

3.5 Diseño de Interacción

3.5.1 Interfaz Orientada al Uso con Superficies Táctiles

La interacción con la herramienta debe de ser lo más transparente posible. Inicialmente se consideró embeber todas las opciones como gestos o controles tangibles. De esta manera el usuario podría trabajar sobre una “hoja de papel inteligente”, libre de botones o elementos de interfaces tradicionales que lo retornarían al contexto de las interfaces más tradicionales. Brevemente, se identificaron las siguientes desventajas de este tipo de implementación:

- El usuario debería de recordar gestos para poder acceder a los comandos del sistema.
- Se añade una carga innecesaria al sistema de reconocimiento de trazos, ya que el mismo debería de diferenciar entre comandos y figuras. Esto se podría convertir en una fuente de

errores inesperados para el usuario.

Como consecuencia, la implementación definitiva de la interfaz presenta tres secciones principales:

- El área de trabajo o “lienzo”. Donde el usuario realizará los trazos y recibirá la retroalimentación del sistema de reconocimiento de trazos.
- Una barra superior con opciones del sistema:
  - Original Strokes / Cleaned Shapes, intercambia entre los modos de visualización de los trazos. El predeterminado – cleaned shapes –, realiza ediciones a los trazos: rectifica líneas y rectángulos, añade leyendas, etc. Original Strokes, mantiene los trazos originales, la única indicación de reconocimiento de un trazo es el color.
  - Export GUI, permite exportar las figuras reconocidas a un formato XML editable en el software de diseño de GUIs, QT Designer.
  - Run, despliega el diseño utilizando Java.
  - Undo y Redo.
  - About, muestra un diálogo con información básica de la aplicación.
  - Exit.

- La barra lateral izquierda contiene herramientas o acciones para manipular el contenido del lienzo:
  - Clear, borra todo el contenido del lienzo.
  - Pencil, La herramienta predeterminada permite dibujar sobre el área de trabajo.
  - Text, facilita el ingreso de texto mediante un teclado en pantalla.
  - Drag, desplaza la posición del lienzo.

Los botones han sido dimensionados apropiadamente para facilitar la interacción no delicada sobre una superficie táctil. Son fácilmente visibles e interpretables, pueden ser accionados con los dedos.

Con el propósito de utilizar puramente la superficie táctil, se implementó un rudimentario teclado en pantalla mostrado en la figura 3.2. De esta manera, el dispositivo que ejecute la herramienta no necesita disponer de un teclado físico. La interfaz del sistema, con todos los componentes descritos en las secciones anteriores se muestra a continuación, en la figura 3.8.

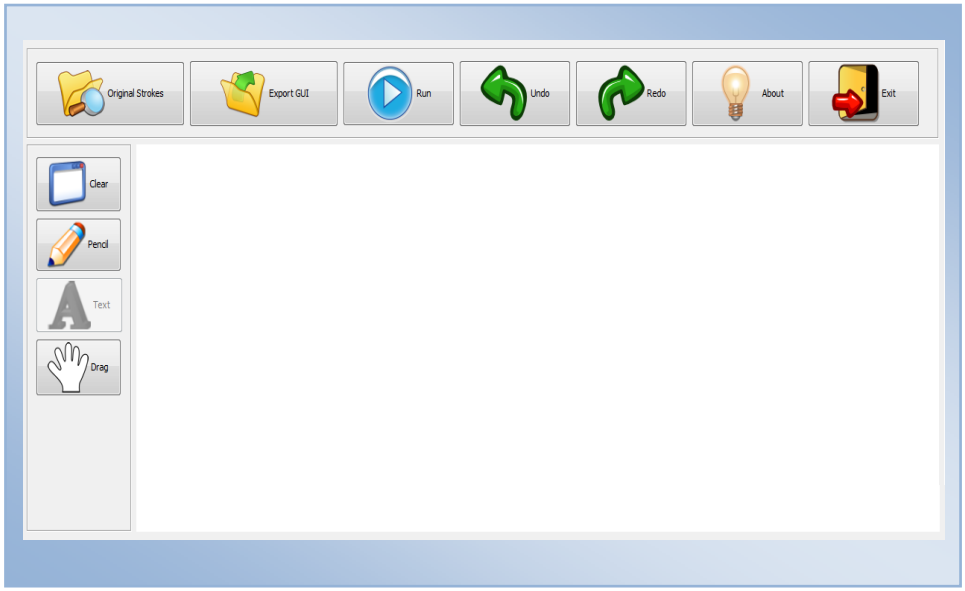


Figura 3.8: Interfaz Gráfica de PenGUIn Prototyper

3.5.2 Retroalimentación Adecuada

Durante la fase de evaluación de sistemas de reconocimiento de trazos, se encontró que es de suma importancia lograr que la transición de trazo a figura reconocida tenga las siguientes características:

- Para el usuario debe ser fácil entender que el sistema ha reconocido un trazo.
- Brindar suficiente información para que el usuario logre discernir si el sistema reconoció con éxito su trazo, o si hizo una interpretación errónea.
- La señal provista por el sistema debe de ser poco intrusiva. La naturaleza de la actividad requiere que esta notificación se

repita con frecuencia, por lo tanto es importante que sea expresada de manera que no distraiga al usuario o imponga acciones innecesarias.

Como compromiso de estos tres objetivos definimos los siguientes estados para un trazo reconocido:

- Líneas.- La unidad más básica para el reconocimiento. El sistema se encargará de mantener información respecto a las propiedades de la línea reconocida. Como se puede observar en la figura 3.9 (a), cuando un trazo es reconocido como una línea, el sistema lo pinta de color azul. Observar también que los trazos no reconocibles aparecen de color rojo.

- Figuras Primitivas.- Las figuras primitivas son los bloques básicos para la formación de figuras más complejas. El sistema les otorga un color rosado, figura 3.9 (b).

- Figuras Complejas o Finales: Son el resultado de combinar más de una figura primitiva o figuras primitivas con líneas. Estas figuras tienen una relación de uno a uno con los “widgets” de la interfaz gráfica. Se dibujan de color negro, ver figura 3.9 (c).

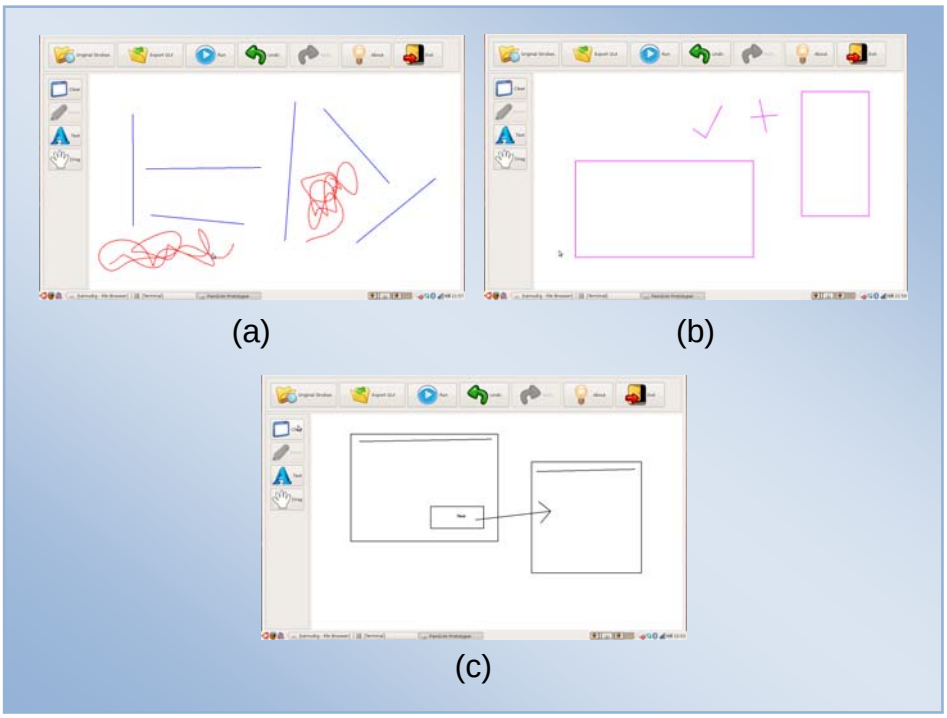


Figura 3.9: Estados de Trazos Reconocidos

La figura 3.10 muestra las diferentes formas en las que se puede ver una interfaz dibujada en PenGUIn Prototyper: Trazos originales 3.10 (a), trazos *limpios* 3.10 (b), apariencia de la interfaz en el modo de ejecución 3.10 (c) y la apariencia de la misma cuando es visualizada en Qt Designer 3.10 (d).

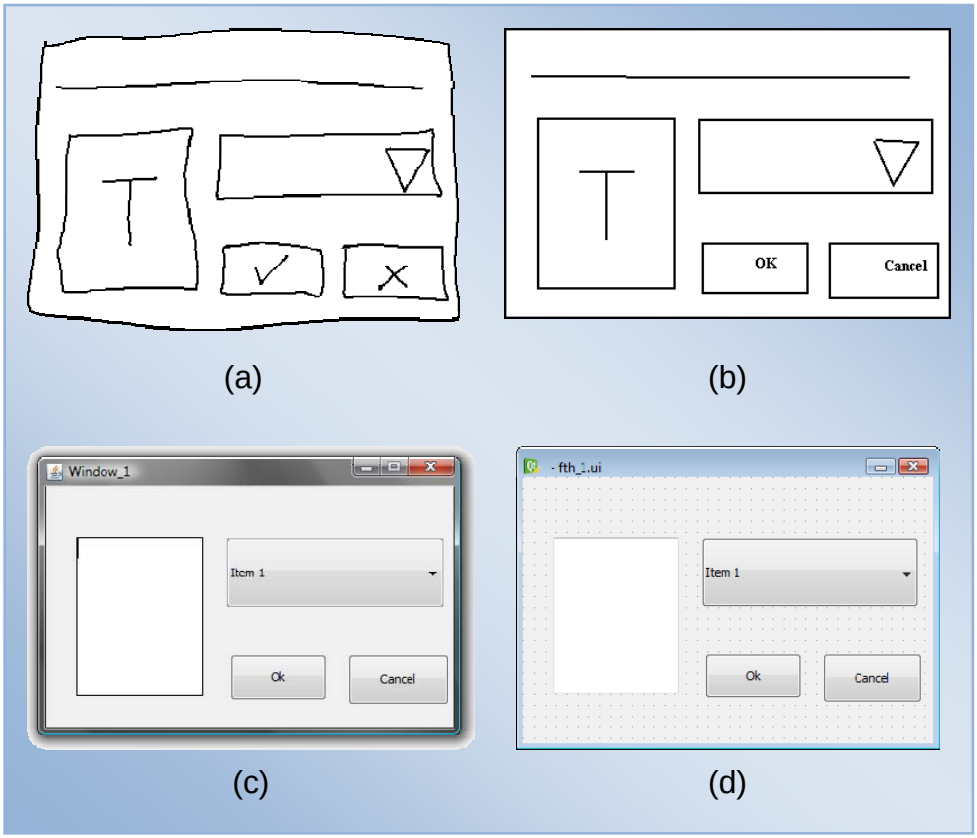


Figura 3.10: Apariencia de una interfaz en PenGUIn Prototyper

3.6 Diseño de Pruebas

Las pruebas sobre el sistema, cuyos resultados se discuten en el siguiente capítulo, fueron realizadas con trece personas de entre 20 y 25 años, cada una de las cuales estaba tomando los cursos de Interacción Hombre-Máquina e Ingeniería de Software II al momento de la ejecución del experimento. En el programa de estudios de cada uno de estos cursos se discuten tópicos sobre la creación de prototipos de interfaces de usuario y varios aspectos relacionados a

su utilidad como medio de evaluación en el proceso de implementación de un sistema.

Las pruebas, realizadas en varias sesiones grupales de hasta cinco personas cada una, eran dirigidas por un instructor que detallaba el procedimiento que debían seguir los participantes y la descripción del escenario en el que iban a colaborar. Una vez dadas las instrucciones, cada participante debía llenar una ficha de datos personales básicos: nombre, edad, género, carrera y nivel de estudio. Esta información era solicitada en un documento de Microsoft Office OneNote 2007 [25] que debía ser contestado a mano alzada con el propósito de introducir al participante en el uso de una pantalla táctil, en el supuesto caso de que éste no haya tenido ninguna experiencia previa con esta clase de dispositivos. Al solicitar esta información, se indicaba al participante que la misma era requerida únicamente con fines de investigación y que su uso en estudios y publicaciones posteriores quedaba bajo la plena consideración de los autores de PenGUIn Prototyper. El documento de datos personales utilizado puede revisarse en el Anexo B.

3.6.1 Ejecución de la Prueba

Después de recibir una rápida introducción acerca del uso del sistema, su modo de interacción, las figuras permitidas en el mismo y

el significado de cada una en el contexto de las interfaces de usuario, se daba inicio a la prueba del sistema, la misma que estaba dividida en dos partes. La primera parte consistía en un experimento guiado donde el participante debía construir un prototipo para un instalador de Winamp [26], un reproductor multimedia utilizado en la plataforma Windows. Este instalador consistía de cinco ventanas, cuyas capturas y la forma en la que estás se vinculaban fueron mostradas a los participantes para que éstos las bosquejen utilizando PenGUIn Prototyper. La figura 3.11 muestra las pantallas expuestas a los participantes en esta parte del experimento. Las flechas de color rojo que parten de un componente en una ventana y llegan a otra indican el flujo de eventos entra ambas interfaces. Las flechas cuyo punto de llegada no es ninguna ventana indican que al activarse dicho control, la ventana actual debe cerrarse.

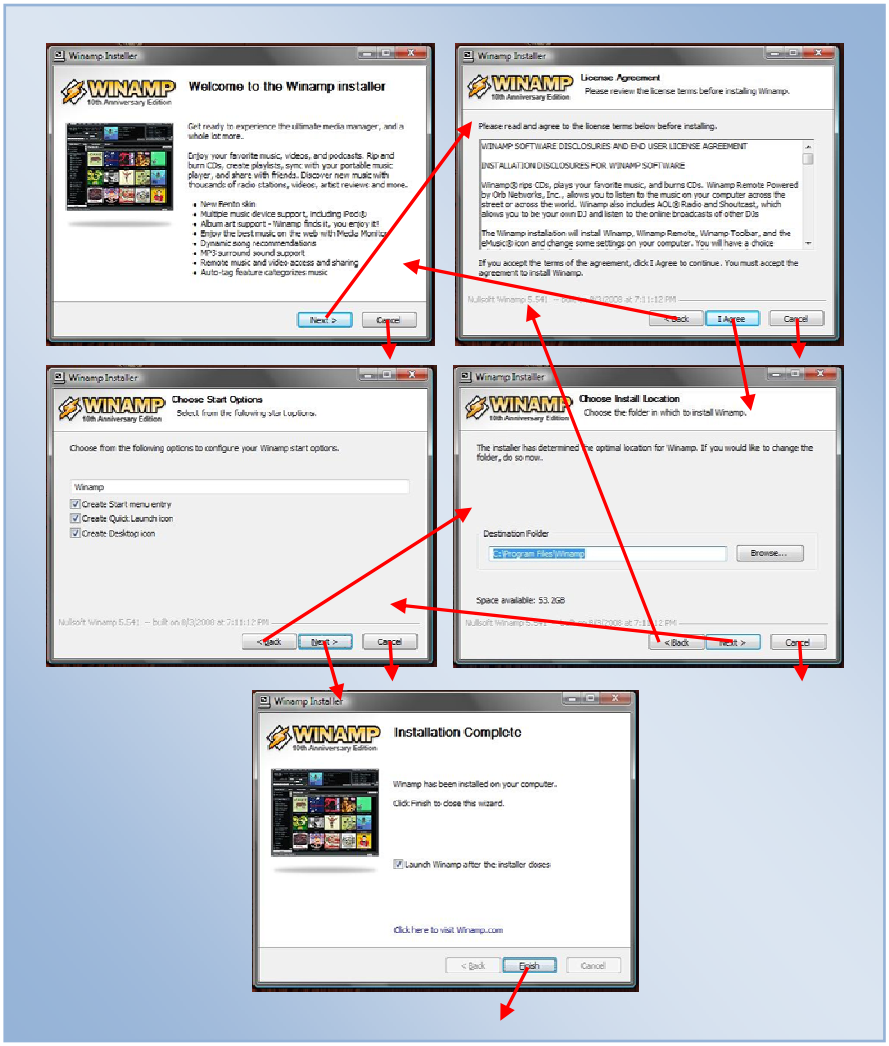


Figura 3.11: Instalador de Winamp de la primera parte de la prueba

La segunda parte, fue un experimento en el que los participantes no recibían instrucciones específicas acerca de la apariencia final que debían tener las interfaces construidas. En esta parte, se solicitaba la construcción de un prototipo para la interfaz gráfica de un sistema para ordenar pizzas. El sistema bosquejado debía cumplir las siguientes restricciones:

- Se debían bosquejar un mínimo de tres ventanas y un máximo de cinco.
- Las interfaces construidas debían tener la apariencia de formularios, es decir, permitir la entrada de datos mediante controles como RadioButtons, CheckBoxes, TextAreas, ComboBoxes y Labels.
- Las interfaces construidas debían utilizar imágenes en sus diseños.
- El prototipo construido debía satisfacer los siguientes requerimientos funcionales del sistema:
  - Existen varios tipos de pizza disponibles para ordenar
  - Es posible ordenar bebidas disponibles en varios sabores.
  - Hay diferentes opciones de pago, entregas, promociones, etc.

Como se ha señalado antes, en este punto de la prueba, cada uno de los participantes era libre de bosquejar sus diseños a su criterio, teniendo en cuenta las restricciones indicadas anteriormente.

3.6.2 Evaluación del Sistema

Una vez finalizado el experimento, se solicitó a los usuarios que

evalúen su experiencia con PenGUIn Prototyper mediante un cuestionario que debía contestarse también a mano, utilizando la Tablet PC. Los criterios evaluados en dicho cuestionario estaban agrupados en tres categorías. Con excepción de la última, que consistía en preguntas abiertas, el usuario calificaba su grado de acuerdo con las proposiciones planteadas en cada sección en una escala de 1 a 5 (del total desacuerdo al total acuerdo). Las categorías evaluadas en el cuestionario utilizado en las pruebas se listan a continuación:

- Información General.- Incluía preguntas referentes a la experiencia previa del usuario con el uso de dispositivos táctiles y de herramientas para el diseño de interfaces de usuario.
- Usabilidad y Facilidad de Aprendizaje.- En esta categoría se evaluó la percepción del usuario acerca de cuán fácil de usar y aprender encontró a la herramienta PenGUIn Prototyper y el grado de satisfacción que tuvo al contrastar sus expectativas con las funcionalidades implementadas en la misma.
- Preguntas Abiertas.- En esta sección se consultaban los aspectos de funcionalidad y apariencia que deberían ser mejorados en PenGUIn Prototyper. Las impresiones finales del

participante al finalizar la prueba eran también consultadas en esta sección.

Con el propósito de tener un registro menos subjetivo del aprendizaje de uso de PenGUIn Prototyper, la salida de la pantalla de la Tablet PC utilizada por cada participante se registró en un archivo a lo largo de toda la prueba. Los videos obtenidos mediante este mecanismo, fueron evaluados a fin calcular el grado de éxito que tuvieron los participantes usando el sistema. Esta medición se la usa para plantear la curva de aprendizaje del sistema.

CAPÍTULO IV.

4 IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se describen los detalles de implementación, tales como la plataforma y los dispositivos de hardware utilizados. Más adelante, se especifican las pruebas de usabilidad y satisfacción del sistema que se realizaron y, en la parte final de este capítulo, se exponen en detalle los resultados de los parámetros medidos en dichas pruebas.

4.1 Plataforma Utilizada

La implementación del sistema fue realizada en el ambiente de desarrollo NetBeans 6.5 utilizando la versión 1.6.0_11 del JVM (Java Virtual Machine) y la 1.6.0_11-b03 del JRE (Java Run-time Environment).

Las versiones (no numeradas) de LADDER y GUILD empleadas para el reconocimiento Multidominio fueron las disponibles en la página del Sketch Recognition Lab [27]. La última modificación del API de LADDER reportada en dicho sitio web fue realizada en Noviembre de 2007 y se presentaba como una publicación no actualizada en su totalidad. A la fecha del último acceso, el sitio anunciaba la pronta

liberación de una nueva y mejorada versión del proyecto LADDER e indicaba que el “release” actual del mismo ya no tenía soporte y que pronto se volvería obsoleto.

Para la representación de las interfaces construidas en un lenguaje que pueda ser interpretado por Qt Designer se utilizó el estándar para representación de datos XML. Los archivos construidos fueron probados en la versión 4.4.3_01 de esta herramienta para diseño de interfaces gráficas.

Dadas las características de portabilidad provistas por Java y XML, la solución implementada es una aplicación multiplataforma. Varias pruebas de PenGUIn Prototyper fueron realizadas en Windows Vista y ambientes Linux (Ubuntu 710).

4.2 Dispositivos de Hardware

La implementación y pruebas del PenGUIn Prototyper se llevaron a cabo utilizando cinco Tablets PC HP Pavilion tx2000. Las vistas mostradas en la figura 4.1 corresponden al uso de este dispositivo como una laptop normal 4.1 (a) o a manera de Tablet PC 4.1 (b).

Figura 4.1: Tablet PC utilizada en la implementación y pruebas

A continuación se listan las principales especificaciones técnicas de este equipo:

- Procesador: AMD Turion 64 X2 TL-58
- 2GB de Memoria RAM
- Dispositivos Inalámbricos: 802.11bgn, Bluetooth.
- Pantalla Wacom con tecnología “Penabled DualTouch”.

La pantalla utilizada permite dos formas de interacción: A través de una pluma digitalizadora provista con la Tablet PC, o con los dedos u otro objeto sólido. Esto permite flexibilidad de uso con los dedos y precisión a través de la pluma para realizar los trazos.

De forma adicional, se realizaron pruebas de ejecución del sistema ligeras utilizando otros dispositivos táctiles:

- Nokia Internet Tablet N800.- Una computadora ultra-portátil con Linux. La resolución y dimensiones bastantes reducidas limitan su utilidad para los fines perseguidos en el presente proyecto.
- Pizarra Digital.- Implementada con el control remoto del Nintendo Wii [28]. Debido a que no se contaba con una instalación permanente de dicha solución, las pruebas realizadas fueron breves y muy pocas.

Dada la alta precisión del dispositivo Wacom, integrado en la Tablet PC, las pruebas sobre los dispositivos antes mencionados, no resultaron muy satisfactorias como para descartar el uso original de la Tablet PC.

4.3 Pruebas

El análisis de las pruebas corroboró las limitaciones previstas del motor de

reconocimiento utilizado. Las pruebas se realizaron con la ayuda de estudiantes de la materia “Interacción Hombre-Máquina” del I término 2008-2009. Un total de 13 estudiantes colaboraron en la sesión.

Se definió el siguiente proceso para la obtención y análisis de los resultados:

1. Se observaron los videos capturados durante las sesiones de prueba para identificar errores y problemas típicos. Dos de estos videos fueron seleccionados para realizar una revisión minuciosa.
2. Los aciertos y errores en la interpretación de trazos fueron registrados con una estampa de tiempo. Los mismos, fueron agrupados en períodos de tiempo de 5 minutos.
3. Se graficaron 3 curvas con respecto a los períodos de tiempo:
   a) Número de Aciertos vs Tiempo.
   b) Número de Errores vs Tiempo.
   c) Eficiencia vs Tiempo. Definimos eficiencia como la diferencia entre aciertos y errores.

4.3.1 Análisis Descriptivo

Se detalla a continuación la prueba del sistema PenGUIn Prototyper, realizada el día 22 de Agosto del 2008, por un estudiante. La misma que demuestra los problemas que encontraron la mayoría de participantes.

El estudiante completó los ejercicios planteados en un tiempo acumulado aproximado de 62 minutos. En las figuras 4.2 y 4.3, las variaciones y picos de errores se atribuyen al proceso de aprendizaje de dibujo de determinadas figuras complejas.

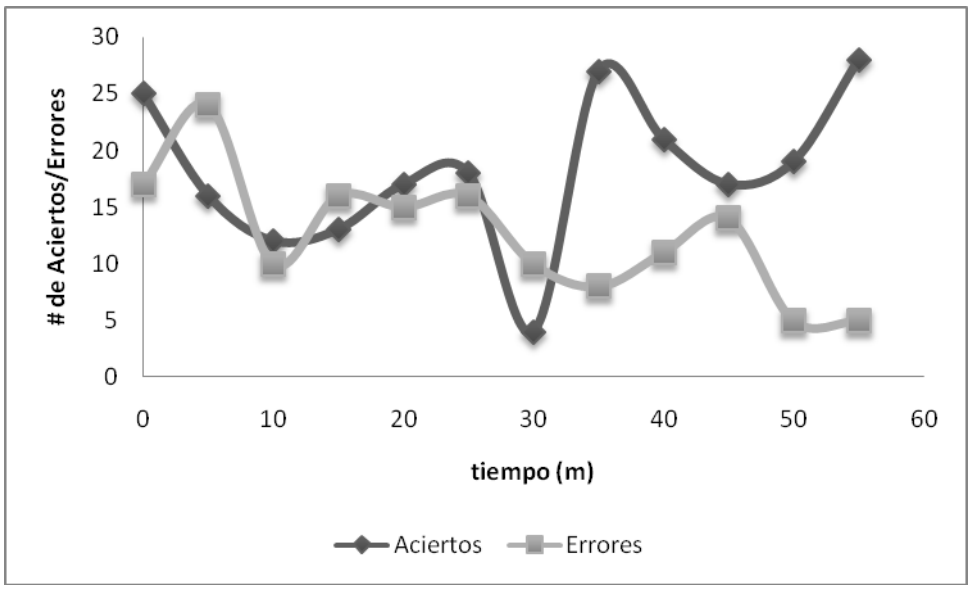


Figura 4.2: Relación entre aciertos y errores en la interpretación de trazos

Los primeros 5 minutos de la sesión se centraron en el mecanismo de prueba y error para aprender a dibujar figuras básicas. Debido a que el usuario tenía problemas logrando que el sistema reconozca rectángulos. El sistema precisa cierta exactitud en las dimensiones de los cuatro lados del rectángulo, así como en los criterios de horizontalidad y verticalidad.

El segundo período, hasta el minuto 10, se caracteriza por la dificultad de

dibujar los botones de siguiente y atrás. El sistema tiene problemas para discernir entre figuras complejas:

- Una ventana está definida como un rectángulo y una barra horizontal de longitud cercana al ancho del rectángulo.
- Un botón de siguiente o atrás, es representado por un rectángulo con una flecha horizontal.

Cuando el usuario comienza a dibujar la flecha para el botón, usualmente hará la barra demasiado larga, y el sistema interpretará esta figura como una ventana. Una vez ocurrido esto, el sistema se niega a reconocer la punta de la flecha, ocasionando que ese trazo sea considerado como un desacierto.

El primer ejercicio transcurre sin mayores problemas desde el minuto 10. En este periodo el usuario dibuja ventanas de diálogo, ingresa texto y ejecuta varias veces el diseño. Alrededor del minuto 30 finaliza la primera parte de la sesión. El gráfico de la figura 4.3 muestra la eficiencia del sistema – que se obtiene al restar el número de errores del número de aciertos detectados en la evaluación –.

Al inicio de la segunda parte, el usuario intenta dibujar “widgets” sobre el lienzo sin haber dibujado antes una ventana. Ya que los descriptores exigen la presencia de una ventana para poder reconocer figuras complejas, el

sistema se niega a interpretar los trazos como botones, áreas de texto y otros. Esto se observa en el pico negativo de eficiencia entre el minuto 30 y el minuto 35.

Cuando el usuario dibuja la primera ventana y soluciona su inconveniente puede realizar el resto de la prueba sin mayores dificultades.

Los problemas alrededor del minuto 45 se deben a la creación de elementos de un formulario. Debido a que los radio buttons y checkboxes están definidos bajo la implementación de proximidad de LADDER, encontramos que con frecuencia el motor agrupa de manera errónea las figuras y etiquetas para formar los respectivos elementos del formulario. Al ser ejecutada la aplicación se observan resultados inesperados.

En los minutos restantes de la prueba se observa que el usuario se ha adaptado más y ha aprendido a tratar con las deficiencias del sistema. En esta fase restante los usuarios se enfocaron en realizar correcciones sobre sus diseños. En los gráficos 4.2 y 4.3, es posible observar que la eficiencia del sistema es mayor, con una baja tasa de error. En los últimos 10 minutos del experimento se registran 47 aciertos contra 10 errores.

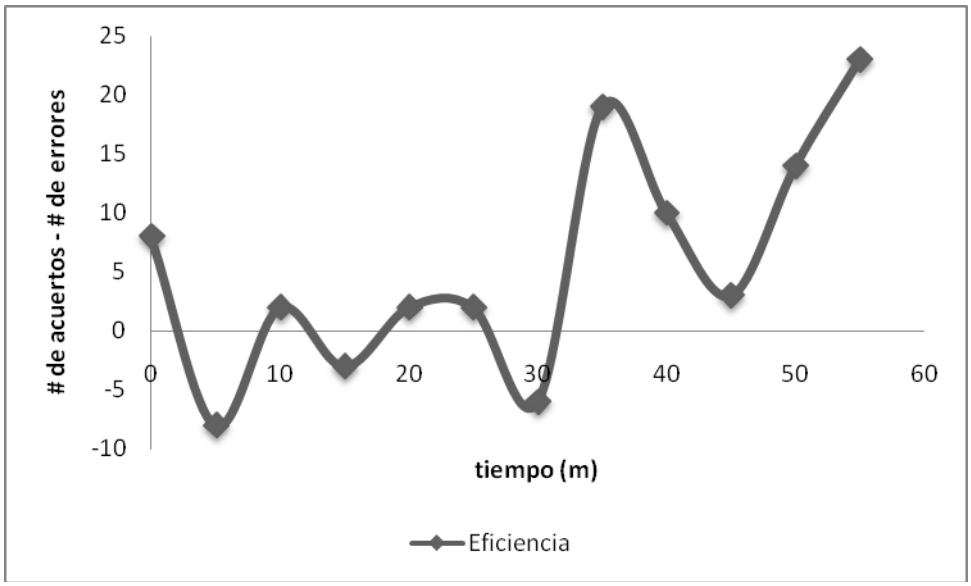


Figura 4.3: Eficiencia del sistema

4.4 Discusión de Resultados

Las evaluaciones del sistema demuestran las limitantes de la tecnología y los algoritmos de reconocimiento de trazos en la actualidad. Debido a la abstracción del papel y la pluma, los usuarios esperan un alto nivel de tolerancia en la aplicación.

Una vez reconocidas las limitantes del sistema, los usuarios fueron capaces de adaptar su estilo de dibujo. De manera que, en subsiguientes trazos, lograron un mayor nivel de eficiencia. En la figura 4.3, se observa que al finalizar el experimento, el usuario ha reducido en gran cantidad la tasa de errores.

Se realizó el mismo análisis para 12 de los 13 participantes. Las figuras 4.4 y 4.5 muestran una generalización del comportamiento descrito en la sección 4.3.1. Estos gráficos fueron obtenidos sumando el número de aciertos y

errores generados por cada uno de los participantes, en su respectivo intervalo de tiempo.

Si bien unos participantes fueron más exitosos en el manejo de la aplicación, los gráficos nos ayudan a observar la tendencia que existió en el experimento.

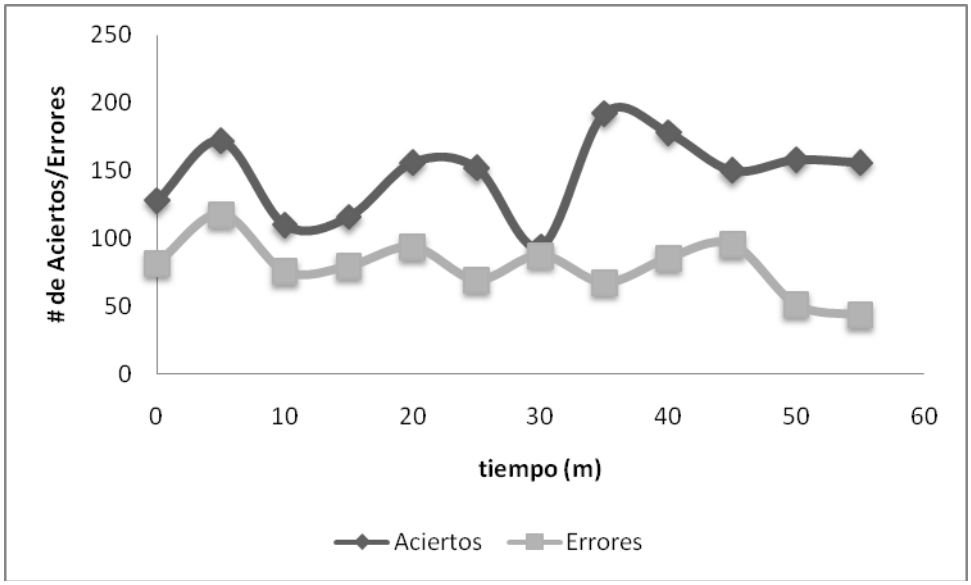


Figura 4.4: Aciertos y Errores acumulados

En la figura 4.5 se observa un declive en la eficiencia alrededor del minuto 30. A pesar de que la efiniciencia es menor que al inicio, se observa que en este punto se paso a la seguida fase de la sesión. Debido a la diferente estructura del ejercicio, la mayoría de los participantes tuvo dificultades en los primeros cinco minutos; sin embargo, los problemas fueron rápidamente superados y se observa el mayor nivel de eficiencia al minuto 35.

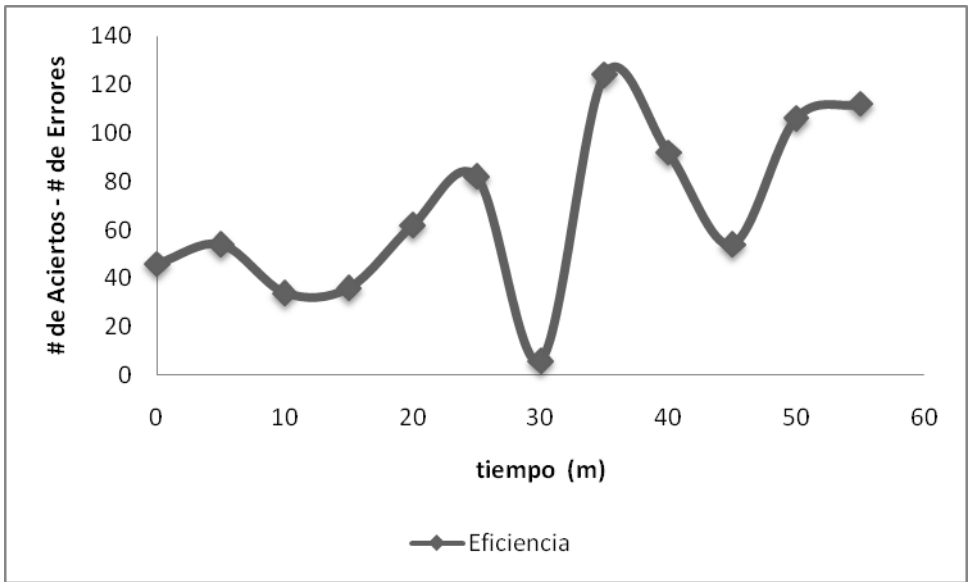


Figura 4.5: Eficiencia acumulada

A continuación se detallan los resultados obtenidos a través del cuestionario de evaluación del sistema. Los hallazgos discutidos corresponden a las secciones de Información General y Usabilidad y Facilidad de Aprendizaje de la evaluación realizada por los participantes en el experimento.

4.4.1 Información general

El 30% de los participantes de la prueba afirmó no haber tenido ninguna experiencia previa en el uso de Tablets PC. El 70% restante afirmó tener poca experiencia y ninguno (0%) de los participantes aseguró ser un usuario experimentado de este tipo de dispositivos. Estos resultados implican que la mayor parte de población de estudio se introdujo en el uso de las Tablets PC con este experimento, por lo que cabe mencionar el hecho de que, al

momento de la prueba, los participantes no habían tenido la oportunidad de desarrollar muchas habilidades de precisión que se adquieren con el uso prolongado de esta clase de dispositivos.

Con respecto a la utilización de herramientas especializadas para el diseño de interfaces de usuario y a la frecuencia con que los participantes las emplean se obtuvieron los siguientes resultados:

Un 22% dijo no haber usado nunca herramientas de este tipo, otro 22% afirmó que rara vez las usan y un 44% dijo emplearlas a menudo. Únicamente en uno de los cuestionarios se afirmó el uso frecuente de herramientas de esta naturaleza. Sin embargo, al detallar los nombres del software utilizado, en este cuestionario específico se listaban los programas Illustrator y Fireworks; cuyo fin es más bien el diseño gráfico y no el de interfaces de usuarios. La lista de los demás nombres de herramientas especializadas en el diseño de interfaces mencionadas incluye a Dreamweaver, cuyo enfoque corresponde al diseño de sitios web, y a otras herramientas menos populares como Infragistics (empleada en el desarrollo de proyectos sobre la plataforma .NET de Microsoft).

4.4.2 Usabilidad y Facilidad de Aprendizaje

Dada la naturaleza cualitativa de las proposiciones planteadas en esta sección del cuestionario, los resultados obtenidos son bastante heterogéneos y son presentados de manera que señalen una tendencia de respuesta

mayoritaria en lugar de valores puntuales o específicos.

Las proposiciones, todas planteadas como observaciones positivas sobre el sistema, debían ser calificadas con valores del intervalo entre 1 y 5, en una escala que representaba el grado conformidad del usuario con las mismas. Así, las respuestas más cercanas a 5 (Total Acuerdo) indican un grado de aceptación de la proposición mayor que las respuestas más cercanas a 1 (Total Desacuerdo).

Los gráficos mostrados en la figura 4.algo ilustran la tendencia de las respuestas de los participantes en cada proposición planteada acerca de PenGUIn Prototyper.

- Facilita el prototipado de interfaces gráficas de usuario

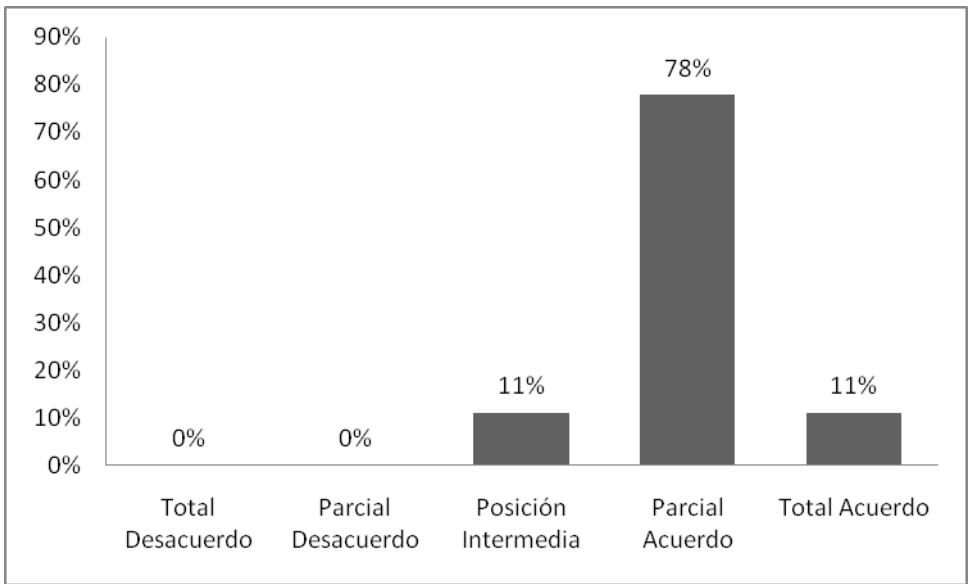


Figura 4.6: Facilidad de Prototipado de GUIs

- Me ahorra tiempo al realizar prototipos o esquemas de GUI's

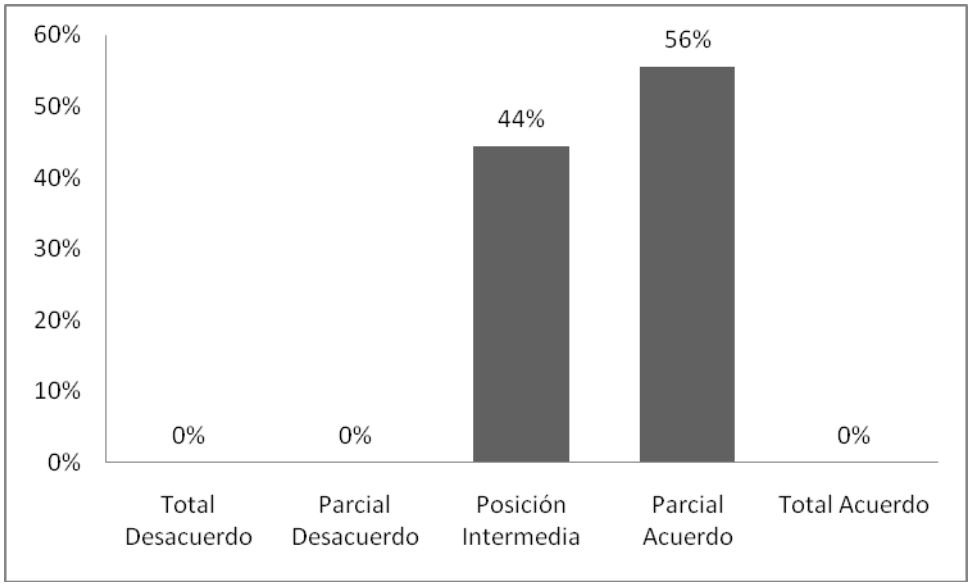


Figura 4.7: Ahorro de Tiempo al Prototipar

- Cumple mis expectativas

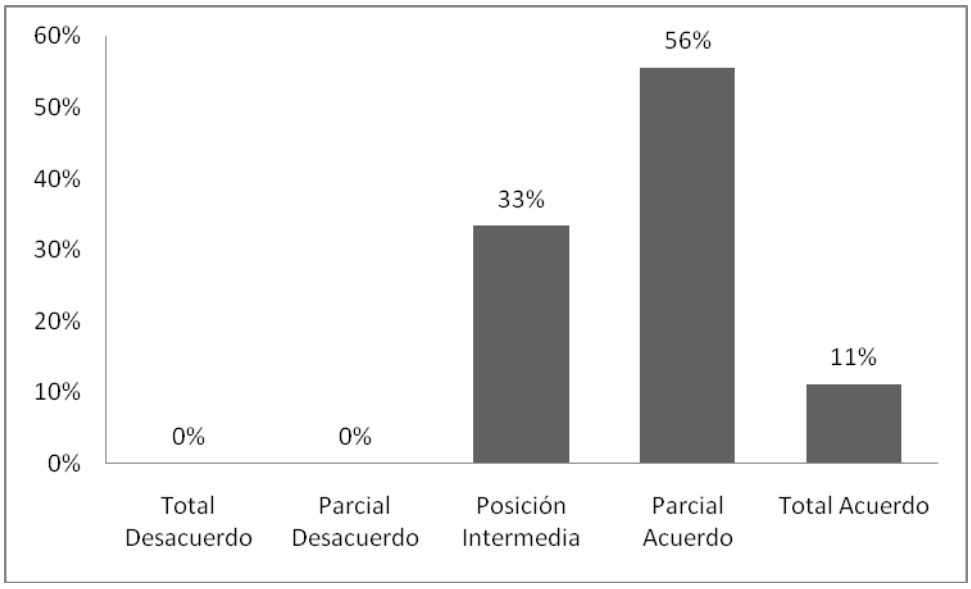


Figura 4.8: Cumplimiento de las Expectativas de los Usuarios

- Es fácil de usar

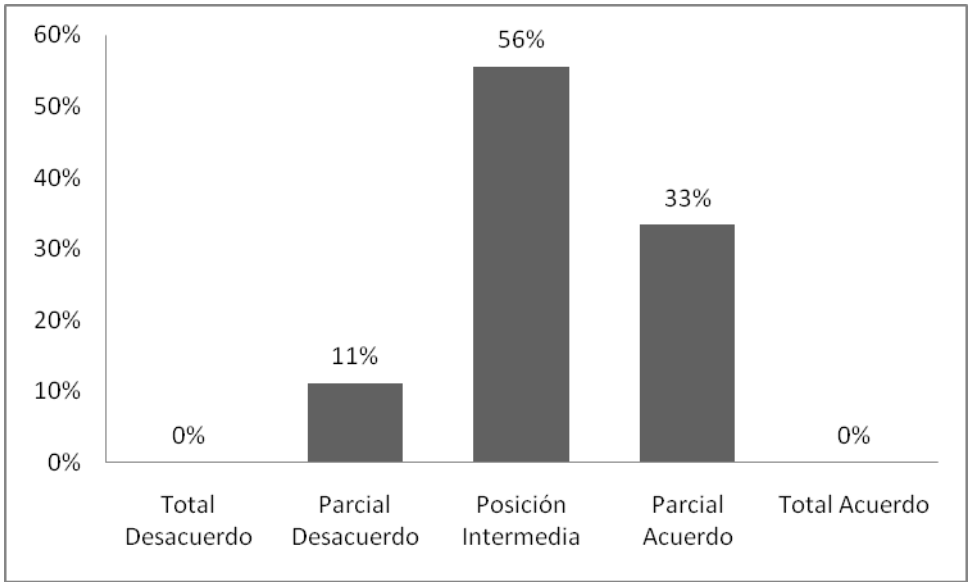


Figura 4.9: Facilidad de Uso

- Se puede usar sin instrucciones previas

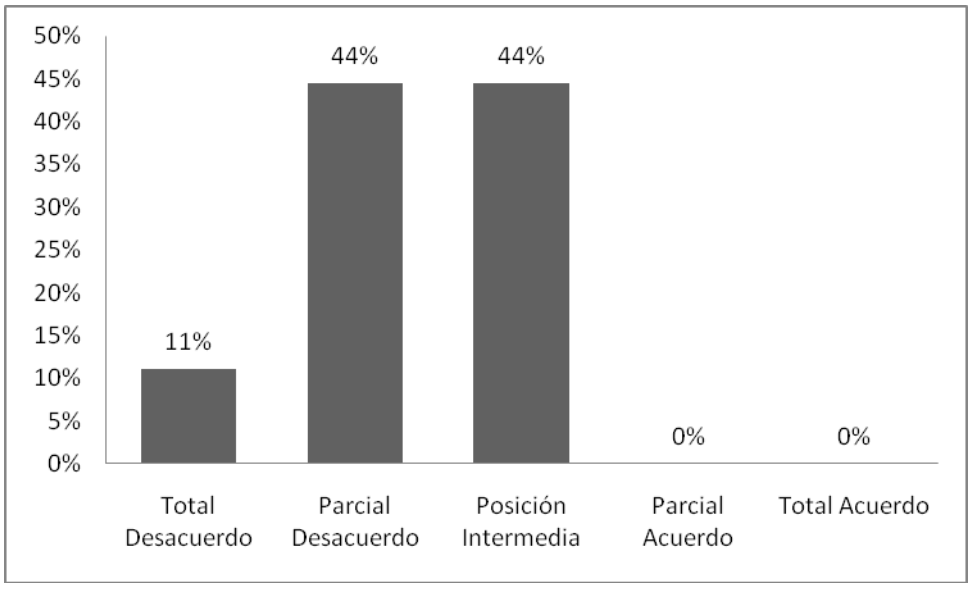


Figura 4.10: Facilidad de Uso sin Instrucciones Previas

Es intuitivo mientras se usa, es decir, no se requiere la ayuda de ningún experto

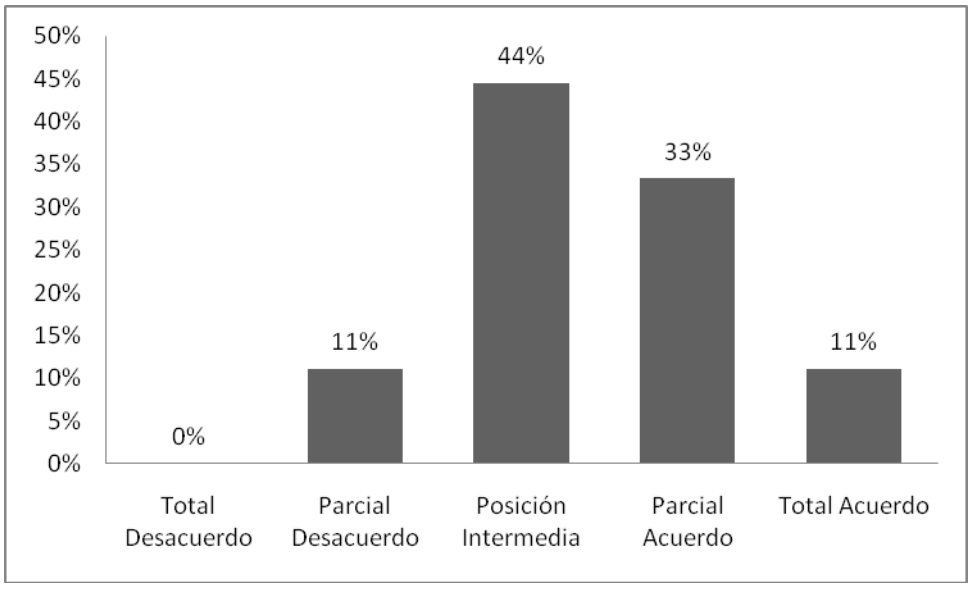


Figura 4.11: Intuitivo al Usar

- Puede ser usado con éxito siempre

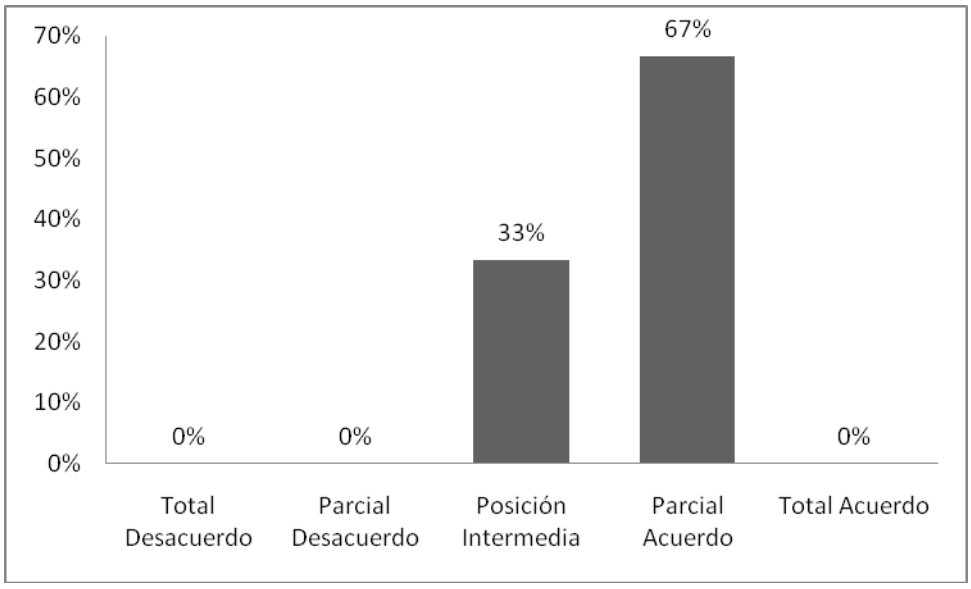


Figura 4.12: Eficacia Percibida

- Aprendí a usarlo rápido

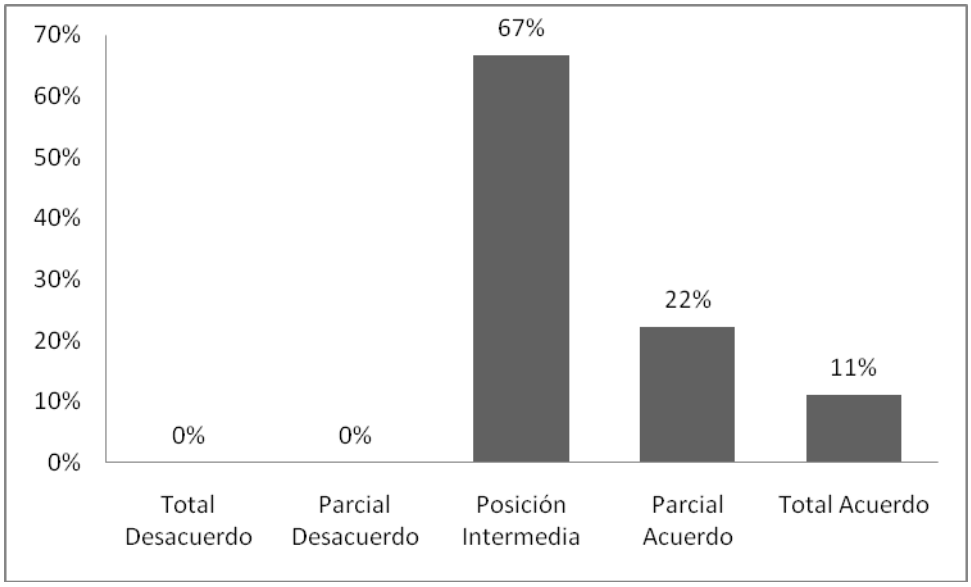


Figura 4.13: Rapidez de Aprendizaje

- Es fácil de aprender a utilizarlo

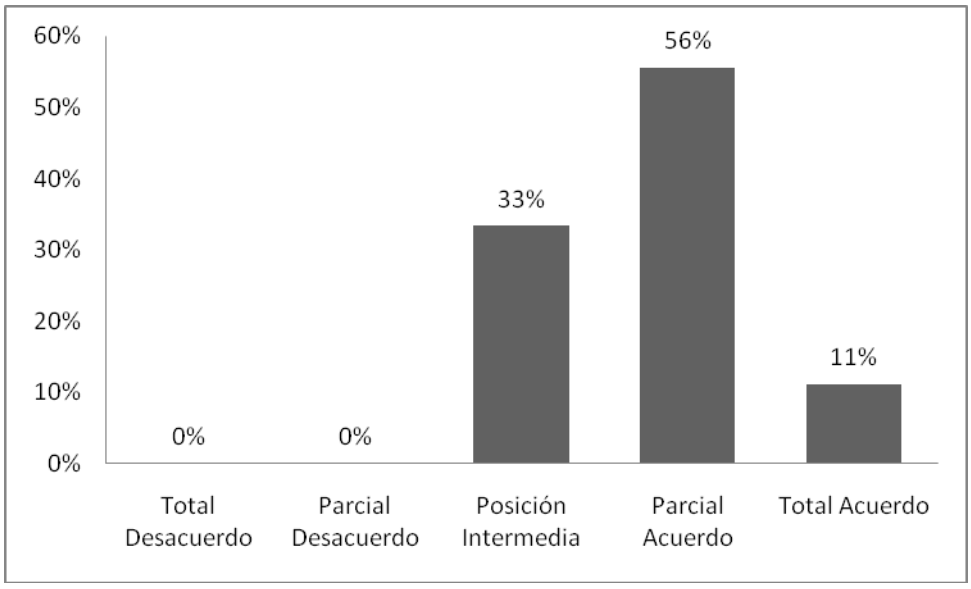


Figura 4.14: Facilidad de Aprendizaje

La sección de preguntas abiertas, acerca de las posibles mejoras o potenciales funcionalidades adicionales, arrojó comentarios que pueden agruparse en las siguientes observaciones generales:

- La mayoría sugiere la redefinición de los descriptores para los botones siguiente y anterior, ya que dibujarlos les resultó, en ocasiones, muy difícil.
- Permitir la personalización de los textos en botones, títulos de ventanas, radiobuttons, checkboxes, cajas de texto, ítems de listboxes y demás controles del sistema.
- Pedir una confirmación al usuario acerca de si realmente desea borrar las interfaces trazadas al presionar el botón “clear”. Al momento, esta sugerencia ha sido ya aceptada e incluida en PenGUIn Prototyper.
- Mejorar el reconocimiento de trazos. Hay trazos que resultan particularmente difíciles de reconocer.

Al consultar sobre los aspectos de la interfaz gráfica de PenGUIn Prototyper, se obtuvieron las siguientes recomendaciones:

- Incorporar ayuda sobre la leyenda de figuras y su representación en el dominio de las interfaces de usuario.
- Permitir opciones de edición tradicional de los trazos realizados: Copiar, pegar, etc.
- Proveer mecanismos que permitan arrastrar los componentes de las

interfaces construidas.

La última sección del cuestionario consultaba al participante sobre sus impresiones finales al participar en el experimento. Esta pregunta arrojó comentarios valiosos que podrían ser considerados como conclusiones aportadas por los participantes después de todo el proceso de prueba y evaluación del sistema. A continuación mostramos algunos de estos comentarios:

> “Muy divertido, realmente te diviertes imaginando mil y un cosas.... Aunque odio el Clear, pero me pareció muy útil para los diseñadores de interfaces. ¡Se les felicita! ”

> “El programa presentado podrá ser de gran ayuda al momento de realizar prototipos, con las mejoras que se le agreguen”

> “Me parece divertida. Aunque algunos dibujos no los reconoce bien, creo que sería de mucha utilidad para diseñar interfaces. La herramienta es muy interesante… Creo que un poco de arreglo sería excelente para el diseño previo de interfaces gráficas”

> “Tiene potencial, pero el reconocimiento de trazos necesita ser

mejorado”

“Me pareció interesante”

La mayoría de estos comentarios son bastante alentadores y demuestran un alto grado de satisfacción al usar el sistema.

CONCLUSIONES

- Los sistemas de reconocimiento de patrones, y específicamente de reconocimiento de trazos libres aún están en desarrollo. La investigación en este campo avanza de manera rápida debido a la alta demanda que existe de aplicaciones con interfaces basadas en el tacto. Sin embargo, la tecnología aplicada en las interfaces actuales está basada en el reconocimiento de gestos, que goza de una mayor eficacia.

- Hicimos contacto con la comunidad científica que desarrolla los “frameworks” que utilizamos. A nuestro conocimiento, este proyecto, a pesar de su limitado alcance es uno de los primeros esfuerzos en nuestro país por desarrollar esta tecnología. Hemos podido ser parte a nivel latinoamericano de discusiones respecto a cómo implementar esta tecnología en las aulas y ambientes colaborativos.

- El reconocimiento de trazos a mano alzada necesita llegar a un estado más refinado para ser útil en aplicaciones reales o de misión crítica.

- A pesar de no tener suficiente evidencia estadística, concluimos que la facilidad de adaptación al sistema es una característica muy

apreciable por los usuarios.

- En este trabajo ha sido posible experimentar con el desarrollo de una aplicación que emplea un paradigma de interacción diferente, debido a la naturaleza de los dispositivos táctiles donde se la ejecuta. Hemos podido dimensionar las consideraciones del diseño de interacción con el usuario, que deben ser observadas al momento de desarrollar aplicaciones de este tipo y de las cuales depende en gran manera el éxito o fracaso de una solución.

- La implementación del sistema utilizando tecnologías libres y abiertas contribuyó en mucha medida a la integración de los módulos construidos con sistemas de reconocimiento multi-dominio, actualmente en desarrollo y permitió que PenGUIn Prototyper pueda ser una herramienta multiplataforma.

RECOMENDACIONES

Esperamos que el diseño e implementación de esta herramienta sea sólo un punto de referencia para nuevas propuestas de interfaces que intenten atacar el problema de reconocimiento de trazos. Consideramos que nuestra solución puede ser mejorada en muchos aspectos: La respuesta del reconocimiento en tiempo real, la precisión con que el sistema evalúa las entradas del los usuarios y el reconocimiento de escritura realizada a mano son algunos de estos aspectos. Con las nuevas y mejores versiones de LADDER y otros sistemas de reconocimiento de trazos, quizá sea posible construir una solución más robusta en el sentido de que sea capaz de lidiar con los errores inherentes a la imprecisión que presentan muchos usuarios al realizar trazos a mano alzada.

- Recomendamos el desarrollo de funcionalidades que puedan extender el uso de PenGUIn Prototyper a situaciones más reales en el ámbito laboral y comercial. La creación de proyectos que puedan ser abiertos directamente en la forma de los trazos originales para su edición y almacenamiento es una característica que haría a esta herramienta muy útil al momento de construir prototipos reales de interfaces gráficas de usuario.

- La implementación de interfaces que logren la comunicación con otras herramientas CASE o CAD serían también de gran utilidad para el sistema. De esta manera, podría lograrse la integración con otras herramientas más avanzadas y así ampliar las funcionalidades provistas en nuestra solución.

- Las pruebas realizadas señalan que aún queda mucho trabajo por hacer. Tratar de minimizar la carga cognitiva del usuario al momento de utilizar la herramienta es probablemente una de las tareas más complejas, pues involucra construir descriptores más sencillos e intuitivos y que sugieran, a pesar de su sencillez, el significado de una figura en el contexto de las interfaces de usuario. Extender el conjunto de los descriptores utilizados en la versión implementada de PenGUIn Prototyper es una mejora que puede lograrse a mediano plazo a fin de incluir otros componentes de interfaces gráficas que no se han considerado en esta solución (scrollbars, grupos de controles, etc.).

REFERENCIAS BIBLIOGRÁFICAS

[1] D. Bäumer et al., “User interface prototyping\—concepts, tools, and experience,” Proceedings of the 18th international conference on Software engineering, Berlin, Germany: IEEE Computer Society, 1996, págs. 532-541; http://portal.acm.org/citation.cfm?id=227841.

[2] C. Kussmaul y R. Jack, “User interface prototyping: tips and techniques,” J. Comput. Small Coll., vol. 21, 2006, págs. 188-190.

[3] C.E. Frank, D. Naugler, y M. Traina, “Teaching user interface prototyping,” J. Comput. Small Coll., vol. 20, 2005, págs. 66-73.

[4] http://www.netbeans.org

[5] http://msdn.microsoft.com/en-us/vstudio/default.aspx

[6] P. J. Molina, S. Martí, and Ó. Pastor, "Prototipado Rápido de Interfaces de Usuario", in Proceedings of the V Workshop Ideroamericano de Ingeniería de Ambientes Software, IDEAS'2002, M. K. e. al.(Eds.), La Habana, Cuba, Abril,2003.

[7] I. Sommerville, Software Engineering: (Update) (8th Edition), Addison Wesley, 2006.

[8] Prototipado y Evaluación: Estudio de casos. [Material gráfico proyectable]. Lleida: Lorés, J. & Granollers, T. 2003. 56 diapositivas.

[9] C. Snyder, Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces, Morgan Kaufmann, 2003.

[10] Sitio oficial de Microsoft Visual Basic. http://msdn.microsoft.com/en-us/vbasic/default.aspx. Último acceso: Enero 28 de 2009.

[11] Sitio oficial de Adobe Dreamweaver. http://www.adobe.com/products/dreamweaver/. Último acceso: Enero 28 del 2009.

[12] W.O. Galitz, The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, Wiley, 2007.

[13] M. Weiser and J.S. Brown, “The coming age of calm technolgy,” Beyond calculation: the next fifty years, Copernicus, 1997, pp. 75-85.

[14] J. Preece, Y. Rogers, y H. Sharp, *Beyond Interaction Design: Beyond Human-Computer Interaction*, John Wiley & Sons, Inc., 2001

[15] H. Ishii, “Tangible bits: beyond pixels,” *Proceedings of the 2nd international conference on Tangible and embedded interaction*, Bonn, Germany: ACM, 2008, pp. xv-xxv.

[16] T. Hammond et al., “Free-sketch recognition: putting the chi in sketching,” *CHI '08 extended abstracts on Human factors in computing systems*, Florence, Italy: ACM, 2008, págs. 3027-3032;

[17] J.A. Landay and B.A. Myers, “Interactive sketching for the early stages of user interface design,” Proceedings of the SIGCHI conference on Human factors in computing systems, Denver,

Colorado, United States: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 43-50.

[18] Keynote Address Implementing a Large-Scale Tablet PC Deployment, The Impact of Tablet PCs and Pen-based Technology on Education, C. Prey, Robert H. Reed and Dave A. Berque. Prudue University Press. Paginas 1-10.

[19] Denim: finding a tighter fit between tools and practice for web site design. In Proceedings of the Conference on Human factors in computing systems, ACM Press, pp. 510–517.

[20] Sitio official de DENIM http://dub.washington.edu:2007/denim/

[21] J.A. Landay and B.A. Myers, Interactive Sketching for the Early Stages of User Interface Design, Carnegie Mellon University, 1994.

[22] J.A. Landay and B.A. Myers, “Sketching Interfaces: Toward More Human Interface Design,” Computer, vol. 34, 2001, pp. 56-64.

[23] Sitio Oficial de Qt Designer. http://www.trolltech.com/products/qt/ Último acceso: Enero 18 de 2009.

[24] T.A. Hammond, “Ladder: a perceptually-based language to simplify sketch recognition user interface development,” Massachusetts Institute of Technology, 2007.

[25] Sitio oficial de Microsoft Office OneNote 2007. http://office.microsoft.com/es-es/onenote/FX100487703082.aspx. Último acceso: Enero 23 de 2009.

[26] Sitio oficial de Winamp en español. http://www.winamp-es.com/- Último acceso: Enero 23 de 2009.

[27] Página web del Sketch Recognition Lab http://srl.csdl.tamu.edu/ladder.shtml Último acceso: Enero 25 de 2009.

[28] Sitio oficial de Nintendo Wii. http://wii.com/ Último acceso: Enero 25 de 2009.

ANEXOS

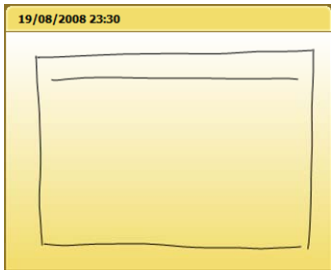
ANEXO A: PRESENTACIÓN PARA LAS PRUEBAS DE USABILIDAD

Agenda

- Descripción de Componentes
- Pruebas:
  - Instalador de Winamp
  - Sistema para venta de pizzas
- Cuestionarios

Componentes

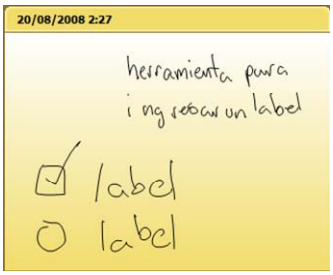
Ventana



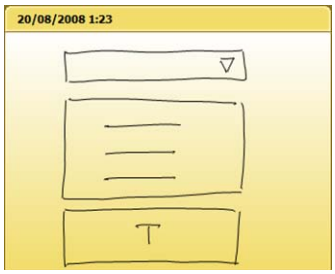
Botones

Componentes

Labels, CheckBox y RadioButtons

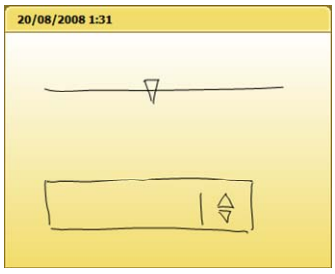


ComboBox, ListBox, TextArea

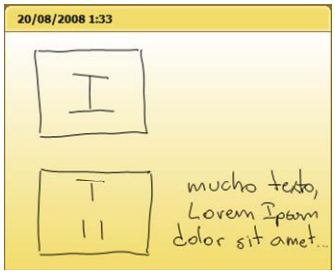


Componentes

Slider y Spinner

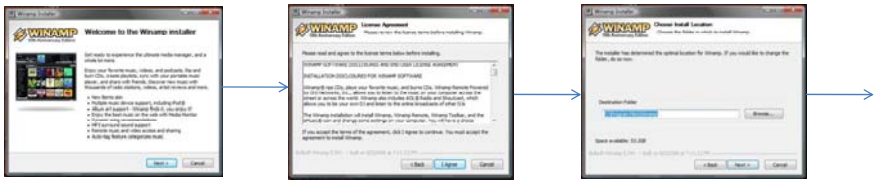


Imágenes y mucho texto

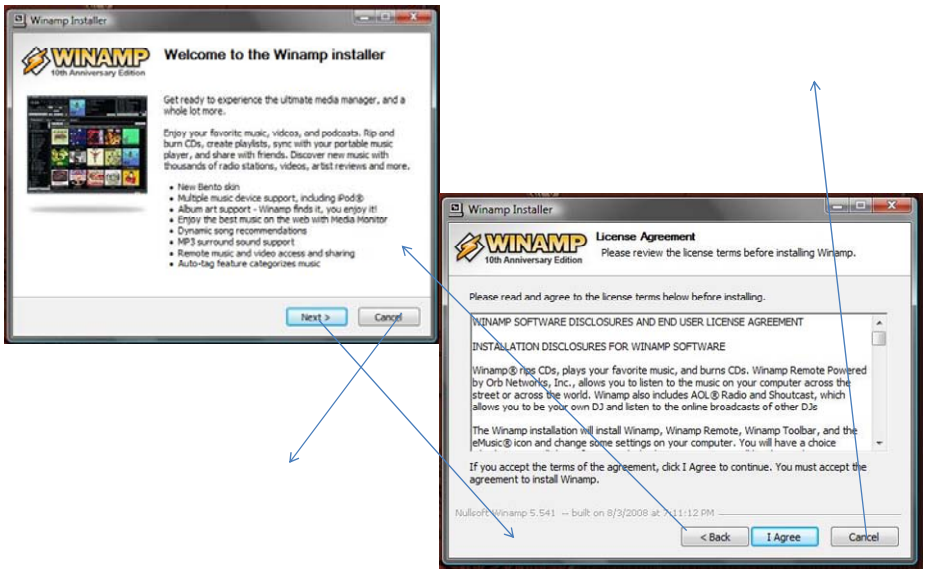


Prueba 1

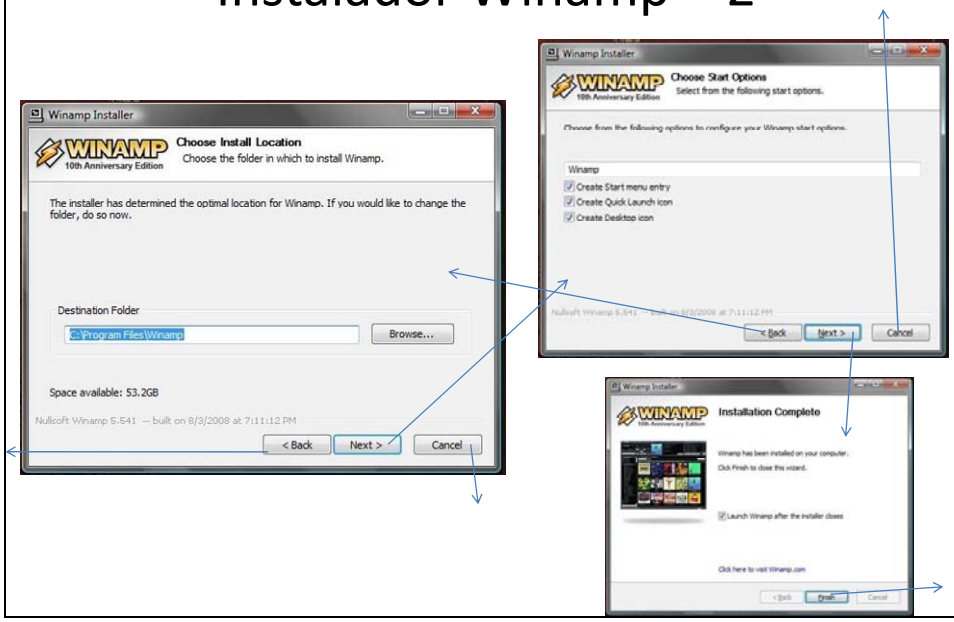
- Dibujar el flujo de pantallas de la instalación del software Winamp.



Instalador Winamp – 1



Instalador Winamp – 2



Prueba 2

- Dibujar la interfaz para un sistema de venta de pizzas. Debe de tener:
  - De 3 a 5 ventanas
  - Formularios (RadioButtons, CheckBoxes, TextAreas, ComboBoxes y Labels)
  - Imágenes
- Considerar varios tipos de pizza, bebidas, opciones de entrega y pago, promociones, etc…

Cuestionario

- Resolver en OneNote

ANEXO B: CUESTIONARIOS DE USABILIDAD

Datos Personales

Nombre:
Edad:
Genero:
Carrera:

Importante:

Toda la información provista en el presente experimento será utilizada con fines investigativos únicamente.

Cuestionario de Evaluación

Sección 1: Información General

1.1 ¿Cuánta experiencia tiene usando Tablets PCs?
Ninguna, Poca, Mucha
1.2 ¿Con qué frecuencia utiliza usted herramientas para el diseño de interfaces de escritorio?
Nunca, Rara vez, A menudo, Siempre
1.3 Califique de 1 a 5 los siguientes enunciados de acuerdo a la escala planteada a continuación:
1 Total Desacuerdo - 5 Total Acuerdo.

Sección 2: Usabilidad

Penguin Prototyper,

2.1 facilita el prototipado de GUIs
2.2 me ahorra tiempo
2.3 cumple con mis expectativas
2.4 es fácil de usar
2.5 se puede usar sin instrucciones previas
2.6 es intuitivo
2.7 puede ser usado con éxito siempre

Sección 3: Facilidad de Aprendizaje

3.1 Aprendí a usarlo rápido
3.2 Es fácil aprender a usarlo

Sección 4: Preguntas Abiertas

4.1 Liste en orden de importancia (de mayor a menor), qué aspectos deben ser mejorados en PenGUIn Prototyper.
4.2 Comente que aspectos de la interfaz de PenGUIn Prototyper deben ser mejorados.
4.3 ¿Cuáles son sus impresiones finales al participar en esta prueba?